Libre como en Libertad

Sam Williams

Aviso Legal

Tabla de contenidos

Traducción al castellano. Comunidad Colibrí

Siguiente
Preface

DT >

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

Anterior
ADDENDUM: How to use this License for your documents

Inicio

Libre como en Libertad:

Inicio

Preface

Tabla de contenidos

The work of Richard M. Stallman literally speaks for itself. From the documented source code to the published papers to the recorded speeches, few people have expressed as much willingness to lay their thoughts and their work on the line.

Such openness--if one can pardon a momentary un-Stallman adjective--is refreshing. After all, we live in a society that treats information, especially personal information, as a valuable commodity. The question quickly arises. Why would anybody want to part with so much information and yet appear to demand nothing in return?

As we shall see in later chapters, Stallman does not part with his words or his work altruistically. Every program, speech, and on-the-record bon mot comes with a price, albeit not the kind of price most people are used to paying.

I bring this up not as a warning, but as an admission. As a person who has spent the last year digging up facts on Stallman's personal history, it's more than a little intimidating going up against the Stallman oeuvre. "Never pick a fight with a man who buys his ink by the barrel," goes the old Mark Twain adage. In the case of Stallman, never attempt the definitive biography of a man who trusts his every thought to the public record.

For the readers who have decided to trust a few hours of their time to exploring this book, I can confidently state that there are facts and quotes in here that one won't find in any Slashdot story or Google search. Gaining access to these facts involves paying a price, however. In the case of the book version, you can pay for these facts the traditional manner, i.e., by purchasing the book. In the case of the electronic versions, you can pay for these facts in the free software manner. Thanks to the folks at O'Reilly & Associates, this book is being distributed under the GNU Free Documentation License, meaning you can help to improve the work or create a personalized version and release that version under the same license.

If you are reading an electronic version and prefer to accept the latter payment option, that is, if you want to improve or expand this book for future readers, I welcome your input. Starting in June, 2002, I will be publishing a bare bones HTML version of the book on the web site, http://www.faifzilla.org. My aim is to update it regularly and expand the [Free as in Freedom] story as events warrant. If you choose to take the latter course, please review Appendix C of this book. It provides a copy of your rights under the GNU Free Documentation License.

For those who just plan to sit back and read, online or elsewhere, I consider your attention an equally valuable form of payment. Don't be surprised, though, if you, too, find yourself looking for other ways to reward the good will that made this work possible.

One final note: this is a work of journalism, but it is also a work of technical documentation. In the process of writing and editing this book, the editors and I have weighed the comments and factual input of various participants in the story, including Richard Stallman himself. We realize there are many technical details in this story that may benefit from additional or refined information. As this book is released under the GFDL, we are accepting patches just like we would with any free software program. Accepted changes will be posted electronically and will eventually be incorporated into future printed versions of this work. If you would like to contribute to the further improvement of this book, you can reach me at mailto:sam@inow.com.

Anterior | Inicio | Siguiente
Libre como en Libertad | Comments and Questions

1. Comments and Questions

Please address comments and questions concerning this book to the publisher:

```
O’Reilly & Associates, Inc.  1005 Gravenstein
Highway North Sebastopol, CA 95472 (800) 998-9938 (in the United
States or Canada) (707) 829-0515 (international/local) (707) 829-0104
(fax)
```

There is a web page for this book, which lists errata, examples, or any additional information. The site also includes a link to a forum where you can discuss the book with the author and other readers. You can access this site at:

```
    http://www.oreilly.com/catalog/freedom/
```

To comment or ask technical questions about this book, send email to:

```
    mailto:bookquestions@oreilly.com
```

For more information about books, conferences, Resource Centers, and the O’Reilly Network, see the O’Reilly web site at:

```
    http://www.oreilly.com
```

2. Acknowledgments

Special thanks to Henning Gutmann for sticking by this book. Special thanks to Aaron Oas for suggesting the idea to Tracy in the first place. Thanks to Laurie Petrycki, Jeffrey Holcomb, and all the others at O’Reilly & Associates. Thanks to Tim O’Reilly for backing this book. Thanks to all the first-draft reviewers: Bruce Perens, Eric Raymond, Eric Allman, Jon Orwant, Julie and Gerald Jay Sussman, Hal Abelson, and Guy Steele. I hope you enjoy this typo-free version. Thanks to Alice Lippman for the interviews, cookies, and photographs. Thanks to my family, Steve, Jane, Tish, and Dave. And finally, last but not least: thanks to Richard Stallman for having the guts and endurance to "show us the code."

Capítulo 1. Todo por una impresora

Temo a los Griegos. Aún cuando traigan regalos.
Virgilio. La Eneida

La nueva impresora estaba atascada, otra vez.

Richard M. Stallman, un miembro de la plana mayor de programadores de software en el Laboratorio de Inteligencia Artificial del Instituto de Tecnología de Massachusetts, descubrió la falla de la manera más dolorosa. Una hora después de enviar un archivo de 50 páginas a la impresora láser, Stallman, de 27 años, interrumpió una sesión productiva de trabajo para recoger sus documentos. Cuando llegó hasta donde estaba la impresora, encontró sólo cuatro páginas en la bandeja de salida de la impresora. Para hacer las cosas aún más frustrantes, las cuatro páginas pertenecían al trabajo de impresión de otra persona, lo cual significaba que el trabajo de impresión de Stallman y la porción sin terminar del trabajo de impresión de alguién más estaban todavía atrapados en alguna parte dentro de la tubería eléctrica de la red de computadores del laboratorio.

El tener que quedarse esperando por las máquinas es uno de los gajes del oficio de los programadores de software, por lo tanto Stallman tomó su frustración con cierta tranquilidad. Además la diferencia entre esperar por una máquina y esperar dentro de una máquina, es apreciable. No era la primera vez en que él era forzado a pararse frente a la impresora a observar las páginas imprimiéndose una tras otra. Siendo una persona que emplea el grueso de sus días y sus noches mejorando la eficiencia de las máquinas y de los programas de software que las controlan, Stallman sintió una urgencia natural de abrir la máquina, mirarle las entrañas, y buscar la raíz del problema.

Desafortunadamente, las destrezas de Stallman como programador de computadoras no se extendían al reino de la ingeniería mecánica. En la medida que documentos recién impresos salían de la máquina, Stallman tuvo una oportunidad de reflexionar en otras formas de solucionar el problema del atascamiento del papel.

Hace cuanto tiempo que los miembros de la plana mayor del Laboratorio de Inteligencia Artificial le habían dado la bienvenida con los brazos abiertos a esta nueva impresora? Stallman se preguntaba. La máquina había sido una donación de la Corporación Xerox. Siendo un prototipo de vanguardia, era una versión modificada de la popular fotocopiadora Xerox. Solo que en lugar de sacar fotocopias, ella dependía de datos enviados por una red de computadoras para convertir estos datos en documentos presentados profesionalmente. Creada por los ingenieros del mundialmente famoso Instituto de Investigación de Palo Alto, era, visto de manera un tanto simple, un adelanto de la revolución en la impresión para los computadores de escritorio que habría de tomarse al resto de la industria de los computadores para los finales de esa década.

Llevados por una urgencia instintiva de jugar con el mejor equipo posible, los programadores del Laboratorio de Inteligencia Artificial prontamente integraron la nueva máquina a la sofisticada infraestructura computacional del laboratorio. El resultado fue inmediatamente placentero. A diferenecia de la antigua impresora láser del laboratorio, la nueva máquina Xerox era rápida. Las páginas volaban a una tasa de una por segundo, convirtiendo a un trabajo de impresión de 20 minutos en uno de tan sólo 2 minutos. La nueva máquina era también precisa. Los círculos eran dibujados

como círculos, no como óvalos. Las líneas rectas parecían líneas rectas, no ondas sinusoidales de baja amplitud.

Era, para todos los propósitos, un regalo demasiado bueno para ser rechazado.

Fue, sólo después de algunas semanas de su llegada que empezaron a hacerse evidentes las fallas de la impresora. La principal entre todas era la susceptibilidad inherente a los atascamientos de papel. Programadores con mente dotada para la ingeniería pronto entendieron la causa detrás los fallos. Igual que una fotocopiadora, la máquina generalmente requería de la directa supervisión de un operador humano. Asumiendo que estos operadores humanos estarían a la mano para arreglar los atascamientos de papel, en caso de que ocurrieran, los ingenieros de Xerox dedicaron su tiempo y sus energías a solucionar otros problemas molestos. En términos ingenieriles, la diligencia del usuario estaba incorporada en el sistema

Al modificar el uso de la impresora, los ingenieros de Xerox habían cambiado la relación usuario-máquina en una manera sutil pero profunda. En lugar de volver a la máquina un subsirviente del operador humano individual, ellos volvieron un subsirviente a toda una población de operadores humanos interconectados a la red. En lugar de tener que pararse frente a la impresora, un usuario humano en un extremo de la red envía su comando de impresión a través de una cadena de máquinas, esperando que el contenido deseado llegue al destino previsto y de la manera apropiada. No fue sino hasta cuando él fue a revisar el resultado final cuando se dió cuenta de lo poco del contenido deseado que había logrado llegar hasta el final.

Stallman mismo había sido de los primeros en identificar el problema y en proponer un remedio. Años atras, cuando el laboratorio todavía usaba su antigua impresora, Stallman había resuelto un problema similar por medio del examen y modificación del programa de software que regulaba la impresora en la máquina PDP-11 del laboratorio. Stallman no pudo eliminar los atascos de papel, pero pudo insertar una instrucción de software que ordenaba a la PDP-11 revisar periódicamente la impresora y reportar este estado de la impresora a la PDP-10, la computadora central del laboratorio. Para asegurar que la negligencia de ningún usuario fuera a poner en peligro la línea completa de trabajos de impresión, Stallman también insertó un comando de software que le instruía a la PDP-10 de notificar a cada usuario si algun trabajo de impresión pendiente que la impresora estaba atascada. El aviso era simple, algo así como "La impresora esta atascada, por favor destrabela," y debido a que iba dirigido a las personas con la necesidad más urgente de arreglar tal problema, las posibilidades eran altas de que el problema fuera solucionado prontamente.

Realizando arreglos, Stallman era evasivo pero elegante. No arreglaba el lado mecánico del problema, pero hacia la siguiente mejor cosa cerrando el bucle de información entre el usuario y la máquina. Gracias a unas pocas líneas adicionales de código de software, los empleados del Laboratorio de Inteligencia Artificial lograron eliminar entre 10 y 15 minutos de tiempo perdido cada semana ejecutando un ir y venir revisando el estado de la impresora. En términos de programación, el arreglo de Stallman tomó ventaja de la inteligencia ampliflicada de toda la red de computadores.

"Si ha usted le llegaba tal mensaje, usted no podía asumir que alguien más iría a arreglar el problema," dice Stallman rememorando la lógica. "usted tenía que ir tras la impresora. Un minuto o dos después de que tuviera problemas, las dos o tres personas que recibieron el mensaje llegaban a arreglar la máquina. De esas dos o tres personas, uno de ellos, al menos, usualmente sabría cómo arreglar el problema."

Tales arreglos ingeniosos eran la marca registrada del Laboratorio de Inteligencia Artificial y de su población nativa de programadores. De hecho, los mejores programadores del Laboratorio de IA desdeñaban el término programador, prefiriendo el título más coloquial de "hacker", en su lugar. Este título cubría una serie de actividades -de todo desde el creativo gozo de mejorar el software existente y los sistemas de computadores. Implícito dentro del título, sin embargo, estaba la anticuada noción de la ingenuidad yanqui. Para ser un hacker, uno tenía que aceptar la filosofía que la escritura de programas de software era sólo el comienzo. Mejorar un programa era la verdadera prueba de las destrezas de un hacker. [1]

Tal filosofía era una razón central por la cual las compañías como Xerox tenían la política de donarles sus máquinas y programas de software a sitios donde típicamente se congregaban hackers. Si los hackers mejoraban el software, las compañías podían pedir prestadas estas mejoras, incorporándolas en versiones actualizadas para el mercado comercial. En términos corporativos, los hackers eran un activo comunitario apalancable, una división auxiliar de investigación y desarrollo disponible a un costo mínimo.

Fue, debido a esta filosofía de toma-y-daca que, cuando Stallman se encontró con ese defecto del atascamiento de papel en la impresora láser, él no entró en pánico. El simplemente buscó la forma de volver a implementar el antiguo arreglo o de "hackear" el nuevo sistema. En el proceso de mirar el software de la impresora láser Xerox, sin embargo, Stallman realizó un descubrimiento problemático. La impresora no tenía ningún software, por lo menos nada que Stallman o sus colegas programadores pudieran leer. Hasta entonces, la mayoría de las compañías tenían como una forma de cortesía el hábito de publicar el código fuente en archivos legibles, los archivos texto que documentaba los comandos individuales de software que le decían lo que debía de hacer la máquina. Xerox, en esta instancia, había provisto archivos de software en forma precompilada, o binaria. Los programadores eran libres de abrir y examinar los archivos si así lo deseaban, pero a menos que fueran unos expertos en decifrar una cadena sinfín de unos y zeros, el texto resultante era pura jerigonza.

Aún cuando Stallman sabía bastante de computadores, él no era un experto en traducir archivos binarios. Siendo un hacker, sin embargo, el tenía otros recursos a su disposición. La noción de compartir información era tan central a la cultura de los hackers que Stallman sabía que era sólo cuestión de tiempo antes de que algún hacker en algún laboratorio universitario o en algún salón de computadores de alguna corporación brindaría una versión del código fuente de la impresora láser, obteniendo así los archivos deseados de código fuente.

Después de los primeros atascos de la impresora, Stallman se confortaba a sí mismo con el recuerdo de una situación similar algunos años antes. El laboratorio necesitabaun programa de red para ayudar a que el PDP-11 trabajara mas eficientemente junto con el PDP-10. Los hackers del laboratorio estaban más que listos para realizar la tarea, pero Stallman, un alumno de Harvard, se recordó de un programa similar escrito por programadores del departamento de ciencias de la computación en Harvard. El laboratorio de computadores de Harvard usaba el mismo modelo de computadores, el PDP-10, pero con un diferente sistema operativo. El computador del laboratorio de Harvard también tenía la política de requerir que todos los programas instalados en el PDP-10 tenían que venir con los archivos de código fuente publicados.

Tomando ventaja de su acceso al laboratorio de computadores de Harvard, Stallman entró al sistema, hizo una copia del código fuente del programa de comunicación de redes, y se lo llevó para el Laboratorio de Inteligencia Artificial. El entonces reescribió el código fuente para hacerlo más adecuado para el sistema operativo del Laboratorio de Inteligencia Artificial. Sin problema y con poca alharaca, el Laboratorio de IA se hizo cargo de una brecha mayor en su infrastructura de software. Stallman aún agregó unas cuantas características que no se encontraban en el programa original de

Harvard, volviendo el programa aún más útil. "Nos las arreglamos usándolo durante varios años," Stallman dice.

Desde la perspectiva de un programador de la era de los 1970s, la transacción fue el equivalente en software de una persona que pide prestada una herramienta o una tasa de azúcar al vecino. La única diferencia era que al tomar prestada la copia de software para el laboratorio de IA, Stallman no había hecho nada para privar a los hackers de Harvard del uso de su programa original. Si acaso algo habían ganado los hackers de Harvard en el proceso, debido a que Stallman había introducido características adicionales al programa, características estas que los hackers de Harvard eran perfectamente libres de tomar prestadas a su vez. Aún cuando nadie en Harvard llegó a pedir prestado el programa modificado, Stallman recuerda a un programador de una firma privada de ingeniería, Bolt, Beranek & Newman, que pidió prestado el programa y le agregó a su vez nuevas características, las cuales Stallman eventualmente habría de reintegrar al archivo del códigos fuente propio del Laboratorio de IA.

"Un programa se desarrolla de la misma forma que una ciudad se desarrolla," dice Stallman, recordando la infraestructura de software del Laboratorio de IA. "Partes son reemplazadas o reconstruidas. Nuevas cosas serán agregadas. Pero usted siempre podrá mirar hacia cierta parte y decir, 'Hmm, por el estilo, veo que esta parte fue escrita a comienzos de los 60s y esta otra parte fue escrita a mediado de los 1970s.'"

A través de este simple sistema de crecimiento intelectual, los hackers en el Laboratorio de IA y de otros sitios construyeron creaciones robustas. En la costa occidental, lo científicos de la computación en la UC de Berkeley, que trabajaban en cooperación con unos pocos ingenieros de bajo nivel en la AT&T, habían construido todo un sistema operativo usando este sistema. Apodado Unix, un juego de palabras hecho a partir de otro sistema operativo más antiguo y más respetable académicamente, llamado Multics, el sistema operativo estaba disponible para cualquier programador dispuesto a pagar el costo de copiar el programa en una nueva cinta magnética y el costo del envío. No todo programador que participaba de esta cultura se describía a si mismo como un hacker, pero muchos de ellos compartían los sentimientos de Richard M. Stallman. Si un programa o una mejora al software era lo suficientemente bueno como para resolver sus problemas, entonces era lo suficientemente bueno como para resolver los problemas de alguien más. Por qué no compartirlo, siguiendo un simple deseo de tener un buen karma?

El hecho que Xerox se hubiera negado a compartir sus archivos de código fuente, al comienzo pareció como una molestia menor. Al empeñarse en buscar una copia de los archivos de código fuente, Stallman dice que él no se preocupó por siquiera contactar a Xerox. "Ellos ya nos habían dado la impresora láser," Stallman dice. "Por qué habría yo de molestarlos con algo más?"

Después de que hubo pasado un tiempo y los deseados archivos no hubieran aparecido, a Stallman le comenzaron a entrar sospechas. El año anterior, Stallman había experimentado un estallido en cólera con un estudiante de doctorado en laUniversidad de Carnegie Mellon. El estudiante, Reid, era el autor de un programa útil de formateo de texto apodado Scribe. Uno de los primeros programas que le dió al usuario el poder de definir fuentes y estilos de tipo cuando se enviaba un documento a través de una red de computadores. El programa era un precursor temprano de HTML, la lingua franca de la World Wide Web. En 1979, Reid tomó la decisión de vender Scribe a una compañía del área de Pittsburgh llamada Unilogic. Finalizando su carrera como estudiante graduado, Reid dice que él simplemente estaba buscando una forma para entregar el programa a un conjunto de desarrolladores que se fueran a encargar de él y así evitar que el programa se deslizara al dominio público. Para endulzar el trato, Reed también acordó insertar un conjunto de funciones dependientes del tiempo - "bombas de tiempo" en la jerga de programadores- este era un dispositivo que desactivaba las versiones copiadas libremente después de un período de expiración de 90 días. Para evitar la desactivación, los usuarios habían de

pagar a la compañía de software, la cual generaba un código que anulaba la bomba de tiempo.

Para Reid, el trato era de ganador-ganador. Scribe no caía en el dominio público y Unilogic se resarcía de su inversión. Para Stallman, era una traición a la ética del programador, pura y simple. En lugar de honrar la noción de compartir, Reid había insertado una forma en que las compañías obligaban a los programadores a pagar por el acceso a la información.

Las semanas pasaron y sus intentos para rastrear el código fuente de la impresora láser encontraron un muro de ladrillos, entonces Stallman comenzó a sentir que se llevaba a cabo un escenario similar de dinero-a-cambio-de-código. Antes de que Stallman pudiera hacer o decir nada al respecto,sin embargo, buenas noticias finalmente gotearon a través del viñedo del programador . Le llegó la voz que un científico del departamento de Ciencias de la Computación de la Universidad de Carnegie Mellon acababa de renunciar a su empleo en el Centro de Investigación de Palo Alto de Xerox. No sólo había este científico trabajado en la impresora láser en cuestión, pero según el rumor, él todavía estaba trabajando en ella como parte de sus tareas de investigación en el Carnegie Mellon.

Poniendo a un lado su sospecha inicial, Stallman se hizo el firme propósito de buscar a la persona en cuestión durante su próxima visita al campus de Carnegie Mellon.

No tuvo que esperar mucho tiempo. El Carnegie Mellon también tenía un laboratorio especializado en investigación de la inteligencia artificial, y en pocos meses, Stallman tuvo una razón relacionada con su trabajo para visitar al campus de Carnegie Mellon. Durante aquella visita, se aseguró de hacer una parada en el departamento de ciencias de la computación. Los empleados del departamento lo encaminaron a la oficina del miembro de la facultad que lideraba el proyecto Xerox. Cuando Stallman llegó a la oficina, encontró al profesor trabajando ahí mismo.

En un estilo clásico de ingeniero a ingeniero , la conversación fue cordial pero directa. Después de presentarse brevemente como un visitante del MIT, Stallman le solicitó una copia del código fuente para la impresora láser para poder portarla al PDP-11. Para su sorpresa, el profesor se negó a cumplir con su solicitud.

"El me dijo que él había prometido el no entregarme ninguna copia," Stallman dice.

La memoria es una cosa graciosa. Veinte años después del suceso, la cinta mental de Stallman acerca de esta historia presenta notoriamente espacios en blanco. No sólo él no recuerda la razón que motivó su viaje ni siquiera la época del año durante la cual realizó tal viaje, el tampoco tiene ningún recuerdo acerca del profesor o estudiante de doctorado que le sirvió de interlocutor. Según Reid, la persona más probable que le negó el código a Stallman es Robert Sproull, un antiguo investigador del Centro de Investigaciones de Palo Alto de Xerox y actual director de Laboratorios Sun, una división de investigación del conglomerado de tecnología de los computadores de Sun Microsystems. Durante los años de 1970s, Sproull había sido el desarrollador primario del software de la impresora láser en cuestión mientras estaba en el Centro de Investigaciones de Palo Alto de Xerox. Alrededor de 1980, Sproull tomó una posición de investigación en la facultad en el Carnegie Mellon donde el continuó su trabajo en la impresora láser entre otros proyectos.

El código que Stallman estaba solicitando era un código de última tecnología que Sproull había escrito más o menos el año anterior al de su llegada a Carnegie Mellon," recuerda Reid. "sospecho que Sproull llevaba menos de un mes en Carnegie Mellon cuando la solicitud de Stallman llegó."

Cuando se le pregunta directamente acerca de esta solicitud, sin embargo, Sproull no afirma ni niega nada. "No puedo hacer un comentario basado en hechos reales," escribe Sproull a través del correo electrónico. "No tengo absolutamente ninguna memoria acerca del incidente."

Con ambos participantes de la corta conversación luchando por recordar detalles claves -incluyendo la de si la conversación siquiera tuvo lugar- es difícil calibrar la rudeza de la negativa de Sproull, al menos como la recuerda Stallman. Cuando se ha dirigido a audiencias, Stallman ha hecho repetidas referencias al incidente, notando que la indisposición de Sproull a entregar el código fuente provino de un *acuerdo de no revelar* (*nondisclosure agreement*), un acuerdo contractual entre Sproull y la Corporación Xerox dándole a Sproull, o cualquier otro signatario, acceso al código fuente del software a cambio de una promesa de mantenerlo en secreto. Actualmente este es un ítem estándar en el negocio de la industria de software, pero el acuerdo de no revelar, o NDA por sus siglas en inglés, era un desarrollo nuevo en esa época, un reflejo de tanto el valor comercial de la impresora láser de Xerox y de la información requerida para manejarla. "Xerox estaba en esa época tratando de hacer de la impresora láser un producto comercial," recuerda Reid. "Habría sido una locura el entregar el código fuente".

Para Stallman, el darse cuenta que Xerox había forzado a su colega programador de participar en este recién inventado sistema de secreto obligatorio le tomó algún tiempo para digerirlo. Al comienzo, todo en lo que pudo enfocarse fue en la naturaleza personal de la negativa. Siendo una persona que se sentía extraño y fuera de sincronía en la mayoría de encuentros cara a cara, el intento de Stallman de presentarse sin anunciarse ante un colega programador era un intento por demostrar buena vecindad. Ahora que la petición había sido denegada, se sentía como si le hubiera cometido una equivocación mayor. "Yo estaba tan furioso, que no encontraba la manera de expresarlo. Así que lo que hice fue retirarme del lugar sin decir ninguna otra palabra," Stallman recuerda. "Yo pude haber tirado la puerta. Quién sabe? Todo lo que recuerdo es que quería irme de ahí inmediatamente."

Veinte años después del hecho, la rabia todavía subsiste, tanto como para que Stallman haya elevado el evento equiparándolo a un punto de quiebre primordial. Dentro de los siguientes meses, una serie de eventos habrían de sobrevenir tanto para Stallman como para la comunidad de hackers del laboratorio de IA que habrían de hacer parecer a los 30 segundos llenos de tensión en una remota oficina de Carnegie Mellon como triviales en comparación. No obstante, cuando llega el momento de ordenar los eventos que habrían de transformar a Stallman, de ser un hacker solitario, instinctivamente sospechoso de la autoridad centralizada, en un activista cruzado que aplica las nociones de libertad, igualdad, y fraternidad al mundo del desarrollo del software, Stallman singulariza de manera especial el encuentro de Carnegie Mellon.

"Me alentó el pensar acerca de algo en lo cual yo había estado pensando," dice Stallman. "Yo ya tenía la idea que el software debía de ser compatido, pero no estaba seguro acerca de como pensar en eso. Mis pensamientos no estaban tan claros ni organizados hasta el punto en que pudiera expresarlos en una manera concisa al resto del mundo."

Aún cuando otros eventos previos habían depertado la ira de Stallman, él dice que no fue sino hasta cuando sucedió el encuentro de Carnegie Mellon que él tomó conciencia que los eventos estaban comenzando a entrometerse en una cultura que él consideraba desde hace mucho tiempo como sacrosanta. Siendo un programador de élite en una de las instituciones élite del mundo, Stallman había perfectamente ignorado los compromisos y transacciones que habían hecho sus colegas programadores, esto en la medida que estas no habían interferido con su propio trabajo. Hasta la llegada de la impresora láser de Xerox, Stallman había estado satisfecho mirando hacia abajo a las máquinas y programas que otros usuarios sombríamente toleraban. En la rara ocasión en que tal programa creó una brecha en las paredes del laboratorio de IA-cuando el laboratorio reemplazó su venerable sistema operativo Incompatible de Tiempo Compartido (*Incompatible Time Sharing* - ITS) con una variante comercial, la TOPS 20, por ejemplo- Stallman y sus colegas hackers habían gozado de la libertad de reescribir, reconstruir y renombrar el software según el gusto personal.

Ahora que la impresora láser se había insinuado en la red del Laboratorio de IA, sin embargo, algo había cambiado. La máquina funcionaba bien, salvo ocasionales atascos de papel, pero la habilidad para modificar según el gusto personal había desaparecido. Desde el viejo punto de vista de toda la industria de software, el caso de la impresora era como un llamado para despertarse. El software se había convertido en una posesión tan valiosa que las compañías dejaron de sentir la necesidad de publicar el código fuente, especialmente cuando la publicación significaba entregar a potenciales competidores la oportunidad de duplicar algo de manera económica. Desde el punto de vista de Stallman, la impresora era una caballo de Troya. Después de una década de fracaso, el software apropiado privadamente -los futuros hackers usarían el término software " propietario" -había ganado una posición firme dentro del Laboratorio de IA a través uno de los métodos más solapados. Había llegado disfrazado de regalo.

El hecho que Xerox hubiera ofrecido a algunos programadores acceso a regalos adicionales a cambio del secreto, también era irritante, pero a Stallman le costó trabajo notar que, si le hubiera sido presentada tal transacción quid pro quo a una edad más joven, el probablemente hubiera aceptado la oferta de la Corporación Xerox. La torpeza del encuentro de Carnegie Mellon, sin embargo, tuvo un efecto afirmador sobre la propia lasitud moral de Stallman. No sólo éste le dió la necesaria rabia para ver todas las futuras súplicas con sospecha, también lo forzó a plantearse la incómoda pregunta: qué pasaría si un colega hacker pasa un día por la oficina de Stallman y es el deber de Stallman el de rehusarse a responderle a la solicitud del hacker de entregar un código fuente?

"Fue mi primer encuentro con un acuerdo de no revelar (NDA) , e inmediatamente me enseño que los acuerdos de no revelar tenían víctimas," dice Stallman, firmemente. "En este caso yo fui una víctima. Mi laboratorio y yo fuimos víctimas."

Fue una lección que Stallman habría de cargar con él a través de los tumultuosos años de 1980s, una década durante la cual muchos de sus colegas de MIT habrían de partir del Laboratorio de AI y habrían de firmar muchos acuerdos de no revelar (NDA) por su propia cuenta. Debido a que muchos de los acuerdos de no revelar (NDAs) tienen fechas de expiración, pocos hackers de los que los firmaron ven poca necesidad de introspección personal. Tarde o temprano, ellos razonaron, el software habría de volverse de público conocimiento. Mientras tanto, el prometer de mantener el software en secreto durante sus primeras etapas de desarrollo era parte de un acuerdo de compromiso que permitía a los hackers el trabajar en los mejores proyectos. Para Stallman, sin embargo, era su primer paso hacia una pendiente resbaladiza.

"Cuando alguién me invita a que traicione a todos mis colegas de esa manera, me recuerdo de lo furioso que me puse cuando alguién más me había hecho eso a mi y a mi propio laboratorio," Stallman dice. "Entonces yo dije, ‘Muchas gracias por ofrecerme este lindo paquete de software pero no puedo aceptarlo en las condiciones que usted me lo ofrece, por lo tanto voy a renunciar a él.’"

Como Stallman habría de aprender rápidamente, el rehusar tales peticiones involucraban algo más que un sacrificio personal. Involucraba el segregarse a si mismo de sus colegas hackers los cuales, aunque compartían un desagrado similar por el secreto, tendían a expresar este disgusto de una manera moralmente más flexible. No fue mucho antes que Stallman, cuando se convertía cada vez más en un paria aún dentro del laboratorio de Inteligencia Artificial, que comenzó a autocalificarse como "el último hacker verdadero", aislándose más y más del mercado dominado por el software propietario. Rehusándole a alguién más el código fuente, Stallman decidió, era no sólo una traición a la misión científica que había nutrido el desarrollo del software desde el final de la Segunda Guerra Mundial, era también una violación de la Regla Dorada, la moraleja que dictaba el no hacer a los demás lo que no querías que los otros te hicieran.

Por lo tanto, la importancia de la impresora láser y el encuentro que surgió de ahí. Sin el, Stallman dice, su vida habría podido seguir un camino más ordinario, uno que balanceara las riquezas de un programador comercial con la frustración final de pasarse la vida escribiendo código de software invisible. No habría habido un sentido de claridad, ninguna urgencia de hacerse cargo de un problema que otros no estuvieran ya haciéndolo. De manera más importante aún, no habría una ira originada en razones virtuosas, una emoción que, como pronto veremos, ha empujado la carrera de Stallman de forma tan segura como cualquier ideología política o creencia ética.

"Desde día en adelante, decidí que esto era algo en lo cual yo nunca participaría," dice Stallman, haciendo alusión a la práctica de intercambiar la libertad personal a cambio de la conveniencia personal -la descripción de Stallman del acuerdo de no revelar -NDA -así como toda la cultura que alentaba ese trato de rasgos tan sospechosos desde un punto de vista ético, en primer lugar. "Decidí nunca convertir a otras personas en víctimas así como yo había sido una víctima."

Notas

[1] Para más sobre el término "hacker," Apéndice B.

Capítulo 2. 2001: La Odisea de un Hacker

El departamento de ciencias de la computación de la Universidad de New York está situado dentro del Hall Warren Weaver, un edificio que parece una fortaleza localizado a dos cuadras al oriente del Washington Square Park. Corrientes de aire acondicionado de fuerza industrial crean un foso de aire caliente, deslentando a holgazanes y a abogados por igual. Los visitantes que logran atravesar el foso encuentran otra formidable barrera, una máquina registradora de seguridad situada inmediatamente adentro del único corredor de entrada de la edificación.

Más allá del punto de registro de seguridad, la atmósfera se relaja en alguna medida. Pero todavía, subsisten numerosas señales desparramadas a lo largo del primer piso que previenen acerca de los peligros de las puertas no aseguradas y de las salidas de incendio *propped-open*. Tomados como un todo, las señales ofrecen un recordatorio: aún los relativamente tranquilos confines de antes del 11 de septiembre del 2001, uno nunca puede ser demasiado cuidadoso o demasiado sospechoso.

Las señales ofrecen un interesante punto de vista contrario al creciente número de visitantes que se agrupan el atrio interior del hall. Unos pocos parecen estudiantes de la Universidad de Nueva York. La mayoría parecen asistentes desgreñados a un concierto haciendo fila en las afueras de una sala de música en aniticipación al acto principal.. Por una breve mañana, las masas se han tomado al Hall Warren Weaver, dejando al guardián de seguridad vecino sin nada mejor que observar a Ricki Lake en television y encoger sus hombros hacia el auditorio contiguo cada vez que los visitantes preguntan por "la charla".

Una vez dentro del auditorio, un visitante encuentra la persona que ha forzado este colapso temporal de los procedimientos de seguridad de la edificación. La persona es Richard M. Stallman, fundador del Proyecto GNU, presidente original de la Fundación del Software Libre (FSF- Free Software Foundation), ganador del programa de Becas MacArthur en 1990, ganador de Premio de Mecánica Computacional de la Asociación Grace Murray Hopper (también en 1990), corecipiente del Premio Takeda 2001 de la Fundación Takeda, y antiguo hacker del Laboratorio de Inteligencia Artificial. Tal como fue anunciado en los sitios web relativos a hackers, incluyendo el sitio del propio Proyecto GNU www.gnu.org, Stallman está en Manhattan, su antiguo hogar, para presentar una muy esperada charla en respuesta a la reciente campaña de la Corporación Microsoft en contra de la Licencia Pública General de GNU.

El tema de la conferencia de Stallman es la historia y el futuro del movimiento de software libre. La ubicación es significativa. Menos de un mes antes, el vice presidente senior de Microsoft Craig Mundie apareció en la vecina Escuela de Negocios Stern de la uNiversidad de Nueva York, presentando una charla en la que soltó una andanada contra Licencia Pública General, o GPL, un dispositivo legal originalmente concebido por Stallman 16 años atrás. Construida para contrarrestar la ola creciente de secretividad en el software, ola que está en proceso de tomarse a la industria del software- una ola que fue notada inicialmente por Stallman durante sus problemas de 1980 troubles con la impresora láser Xerox laser -la GPL ha evolucionado hacia una herramienta central de la comunidad del software libre. En términos más simples, la GPL encierra los programas de software en una forma de propiedad comunal-lo que hoy los eruditos legales llaman los "comunes digitales" ("digital commons") -a través del peso legal del copyright. Una vez encerrados, los programas permanecen inamovibles. Versiones derivadas deben llevar la misma protección del copyright-aún

versiones derivadas que contienen sólo una pequeña porción del código fuente original. Por esta razón, algunos dentro de la industria del software han dado en llamar a la GPL como una licencia "viral", pues esta se disemina a todos los programas de software que toca. [1]

En una economía de información crecientemente dependiente del software y agradecida de los estándares de software, la GPL se ha convertido en el proverbial "gran garrote". Aún las compañías que una vez se rieron de ella tachándola de socialismo de software se han juntando alrededor reconociendo los beneficios. Linux, el núcleo similar a Unix desarrollado por el estudiante universitario finés Linus Torvalds en 1991, está licenciado bajo la GPL, así como lo están muchas de las más populares herramientas de programación: GNU Emacs, el GNU Debugger, el Compilador GNU C, etc. Conjuntamente, estas herramientas forman los componentes de un sistema operativo de software libre desarrollado, mantenido y perteneciente a una comunidad de hackers localizada en diferentes partes del mundo. En lugar de ver esta comunidad como una amenaza, las compañías de alta tecnología como IBM, Hewlett Packard, y Sun Microsystems han llegado a depender de ella, vendiendo aplicaciones de software y servicios construidos sobre la siempre creciente infraestructura de software libre.

Ellos han llegado también a depender de esta infraestructura, haciendo uso de esta como arma estratégica en la guerra perenne de la comunidad hacker contra Microsoft, aquella compañía que tiene su base en Redmond, Washington- y que, para bien o para mal, ha dominado el mercado de software para PCCs desde finales de los 1980s. Como dueño del popular sistema operativo de Windows, Microsoft es quien puede tener las mayores pérdidas en este cambio en la industria del software hacia la licencia GPL. Casi toda línea del código fuente del coloso de Windows esta protegida por derechos reservados de autor reafirmando la naturaleza privada del código fuente subyacente o, al menos, reafirmando la habilidad legal de Microsoft de tratar al código fuente como una propiedad privada. Desde el punto de vista de MIcrosoft, incorporar programas protegidos por la licencia "viral" GPL dentro del colosos de Windows equivaldría a que Superman se tomara unas pastillas de Kriptonita. Compañías rivales podrían de pronto copiar, modificar, y vender versiones mejoradas de Windows, transformando instantáneamente la posición dominante de la compañía como proveedor No. 1 de software orientado al consumidor en una posición vulnerable. Por lo tanto, de ahí viene la creciente preocupación de la compañía por la tasa de adopción de la GPL. Y esta es la causa de la reciente charla de Mundie en que soltó la andanada contra la GPL y el abordaje de "fuente abierta" en lo referente al desarrollo de software y las ventas. De donde provino la decisión de Stallman de presentar una refutación pública a tal conferencia en el mismo campus y el día de hoy.

20 años es mucho tiempo para la industria del software. Considere esto: en 1980, cuando Richard Stallman estaba maldiciendo la impresora láser Xerox del Laboratorio de IA, Microsoft, la compañía que los hackers modernos ven como la fuerza más poderosa en la industria del software en todo el mundo, era todavía una principiante. IBM, la compañía que los hackers usualmente miraban como la fuerza más poderosa dentro de la industria del hardware en todo el mundo, todavía estaba por introducir el computador personal, y entonces inaugurando el actual mercadoo de PCs de bajo costo. Munchas de las tecnologías que ahora tomamos por dadas -el World Wide Web, la televisión satelital, las consolas de video-juegos de 32-bit-ni siquiera existían. Lo mismo va por lo que tiene que ver con las compañías que ahora llenan los escalones de arriba del establecimiento corporativo, compañías como AOL, Sun Microsystems, Amazon.com, Compaq, y Dell. La lista se prolonga aún más.

El hecho que el mercado de alta tecnología haya llegado tan lejos en tan poco tiempo es combustible para ambos lados del debate entorno de la licencia GPL. Los proponentes de la GPL apuntan a la corta vida de muchas de las plataformas de hardware de computador. Enfrentando el riesgo de comprar un producto obsoleto, los consumidores tienden a conglomerarse en torno a compañías con mayor sobrevivencia a largo término. Como resultado de esto, el mercado de software se ha convertido en un

campo de batalla donde el ganador lo toma todo. [2] El actual, ambiente de software apropiado privadamente, los proponentes del GPL dicen, lleva al abuso del monopolio y al estancamiento. Fuertes compañías se chupan todo el oxígeno del mercado quitándoselo a los competidores rivales y a los principiantes innovadores.

Los oponentes al GPL argumentan exactamente lo opuesto. La venta de software es tan riesgosa, sino más riesgosa que, la compra de software, ellos dicen. Sin las garantías legales provistas por las licencias de software privadas, para no mencionar los prospectos económicos de una "aplicación matadora" apropiada privadamente (i.e., una tecnología de punta que dispara un mercado enteramente nuevo), [3] las compañías pierden el incentivo a participar. De nuevo, el mercado se estanca y la innovación declina. tal como Mundie puso en relieve en su charla deñ 3 de mayo en el mismo campus, la naturaleza "viral" de la licencia GPL "presenta una amenaza" contra cualquier compañía que dependa de la unicidad de su software como activo de competitividad. Mundie agregó:

> Esta también socava fundamentalmente el sector de software comercial independiente debido a que efectivamente hace que sea imposible el distribuir software bajo una base donde los compradores pagan por el producto algo más que tan sólo el costo de la distribución. [4]

El éxito mutuo de GNU/Linux, el sistema operativo amalgamado contruido alrrededor del núcleo Linux protegido por la GPL, y de Windows a lo largo de los últimos 10 años revela la sabiduría de ambas perspectivas. Sin embargo, la batalla por el impulso es muy importante en la industria del software. Aún poderosos vendedores tales como Microsoft dependen del apoyo de terceros desarrolladores cuyas herramientas, programas, y juegos de computadores hacen a las plataformas de software subyacentes tales como Windows, más atractivas para el consumidor promedio. Citando la rápida evolución del mercado de tecnología a lo largo de los últimos 20 años,para no mencionar el admirable registro de su propia compañía durante este último período, Mundie aconsejó a sus escuchas de no dejarse influenciar demasiado por el impulso reciente del movimiento de sofwtare libre:

> Dos décadas de experiencia han mostrado que un modelo económico que protege la propiedad intelectual y un modelo de negocios que recorta los costos de investigación y desarrollo pueden crear beneficios económicos impresionantes y también puede distribuir estos beneficios ampliamente. [5]

Tales advertencias sirven como telón de fondo para la charla de Stallman del día de hoy. Menos de un mes depués de su declaración, Stallman permace erguido con su espalda hacia uno de los tableros al frente del cuarto, inquieto por comenzar.

Si las últimas dos décadas han traido dramáticos cambios al mercado del software, ellos han traido aún más dramáticos cambios para Stallman mismo. Ya no está el hacker delgado, afeitado limpiamente que una vez empleó por entero sus días comunicándose con su amada PDP-10. En su lugar está un hombre pesado de mediana edad con largos cabellos y barba de rabino, un hombre que ahora emplea el grueso de sus días escribiendo y respondiendo emails, arengando a sus colegas programadores, y dando charlas como las de hoy. Vestido con una camiseta *aqua-colored* y pantalones carmelitos de polyester, Stallman parece un ermitaño del desierto recién salido del vestier del Ejército de Salvación.

El auditorio esta lleno de visitantes que comparten la moda y los *grooming* gustos de Stallman. Muchos llegan portando computadores portátiles y modems celulares, tanto mejor para registrar y transmitir las palabras de Stallman a una ansiosa audiencia de internet. El porcentaje de hombres con respecto a las mujeres es de unos 15 hombres por cada mujer, y una de cada 7 u 8 mujeres porta un muñeco de pinguino, la mascota oficial de Linux, mientras que otras llevan un osito de peluche.

Agitado, Stallman abandona su puesto al frente del salón y toma un asiento de la primera fila, digitando unos pocos comandos en un computador portátil previamente abierto. Por los siguientes 10 minutos Stallman está abstraido del creceinte número de estudiantes, profesores, y fanáticos que circulan en frente de él al pié de la tarima del auditorio.

Antes de que la charla pueda comenzar los rituales barrocos de la formalidad académica deben de ser observados. La aparición de Stallman merece no una sino dos presentaciones. Mike Uretsky, codirector del Centro de Tecnología Avanzada de la Escuela Stern, provee la primera.

"El papel de una universidad es el de fomentar el debate y de tener interesantes discusiones," dice Uretsky. "Esta presentación particular, este seminario casa perfectamente en ese molde. Encuentro la discusión acerca de las Fuentes Abiertas particularmente interesante."

Antes de que Uretsky pueda comenzar una nueva frase, Stallman está de pié manoteándole como un motorista varado.

"Yo hago software libre," Stallman dice ante las carcajadas crecientes. "Fuentes Abiertas es un movimiento diferente."

La risa da paso al aplauso. El cuarto esta repleto de partidarios de Stallman, personas que saben de su reputación por la exactitud verbal, para no mencionar su muy publicitada controversia con los proponentes de movimiento de Fuentes Abiertas. Muchos han venido para anticipar tales explosiones del mismo modo en que anteriormente los fanáticos de la radio aguardaban por la marca registrada de Jack Benny, la exclamación de "Cortala!" en cada uno de sus programas de radio.

Precipitadamente Uretsky finaliza su presentación y cede el escenario a Edmon Schonberg, un profesor del departamento de ciencias de la computación de la Universidad de Nueva York. Como programador de computadores y colaborador del proyecto GNU, Schonberg conoce las minas explosivas linguísticas que debe de evitar. Diestramente sintetiza la carrera de Stallman desde la perspectiva de un programador de hoy en día.

"Richard es el ejemplo perfecto de alguien que, actuando localmente, comenzó a pensar globalmente [acerca] de los problemas que conciernen a la no disponilbilidad del código fuente," dice Schonberg. "El ha desarrollado una filosofía coherente que nos ha forzado a todos nosotros a reexaminar nuestras ideas acerca de como el software es producido, de lo que la propiedad intelectual significa, y de lo que la comunidad de software representa de hecho."

Schonberg da la bienvenida a Stallman con más aplausos. Stallman se toma un momento para cerrar su computador portátil, se levanta, y se toma el escenario.

Inicialmente, las palabras de Stallman parecen más una rutina de comedia que un discurso político. "Quisiera agradecer a Microsoft por darme la oportunidad de aparecer en esta plataforma," Stallman hace el comentario agudo. "Durante las últimas semanas, me he sentido como un escritor cuyo libro hubiera sido fortuitamente prohibido en alguna parte."

Para los no-iniciados, Stallman se sumerge en una pronta analogía de calentamiento acerca del software libre. El asemeja un programa de software a una receta de cocina. Ambos proveen unas útiles instrucciones paso a paso de como completar una tarea deseada y pueden ser fácilmente modificadas si un usuario tiene deseos especiales o circustancias. "Usted no tiene que seguir la receta al pie de la letra," Stallman anota. "Usted puede prescindir de algunos ingredientes. Agregar algunos hongos, ’pues a usted le gustan los hongos. Poner menos sal pues su médico le recomendó que le bajara a la sal- o lo que sea."

De manera más importante, Stallman dice, los programas y las recetas son ambas fáciles de compartir. Al dar una receta a un invitado a cenar, el cocinero pierde solo un poquito de tiempo y el costo del papel en el que la receta se escribe. Los programas de sofwtare requieren aún menos, usualmente unos cuantos clicks del mouse y un poquito de electricidad. En ambas instancias, sin embargo, la persona que da la información gana dos cosas: una amistad incrementada y la habilidad para pedir prestado interesantes recetas como intercambio.

"Imaginense lo que sería si las recetas vinieran empacadas en cajas negras," Stallamn dice, cambiando la velocidad. "Usted no podría saber los ingredientes que usan, tampoco podría cambiarlos, e imaginense que pasaría si usted hace una copia para un amigo. Usted sería acusado de pirata y tratarían de ponerlo en prisión por años. Ese mundo atropellaría tremendamente a todo aquel que tuviera la costumbre de compartir cosas con los demas. Pero así es exactamente el mundo del software propietario. Un mundo en el que la decencia hacia otras personas esta prohibida o es evitada."

Presentada esta analogía introductoria, Stallman se lanza a contar el episodio de la impresora láser de Xerox. Igual que con la analogía de la receta, la historia de la impresora láser resulta ser un dispositivo retórico útil. Con su estructura que se asemeja al de una parábola, esta dramatiza acerca de que tan rápidamente las cosas pueden cambiar en el mundo del software. Llevando a sus oyentes a una época antes de la compra con un solo click de Amazon.com, de Microsoft Windows, y de las bases de datos de Oracle, él lleva al escucha a examinar la noción de propiedad del software liberado de los logos corporativos actuales.

Stallamn presenta la historia con todo el pulimiento y la práctica de un fiscal de distrito local que conduce un argumento de cierre. Cuando llega a la parte en que el profesor de Carnegie Mellon se rehusa a entregarle una copia del código fuente de la impresora , Stallman hace una pausa.

"El nos ha traicionado," Stallman dice. "Pero el no solo nos lo hizo a nosotros. ES bien probable que también lo traicionó a usted."

En la palabra "usted," Stallman señala con su dedo índice acusador a un insospechado miembro de la audiencia. Las cejas de este miembro de la audiencia titubean levemente, pero los propios ojos de Stallman se han movido. Lenta y deliberadamente, Stallman escoge a un segundo miembro de la audiencia que rie disimulada y nerviosamente. "Y creo, que muy probablemente, se lo hizo a usted, también," el dice, señalando a un miembro de la audiencia que esta tres filas detrás del primero.

Para cuando Stallman ha escogido a un tercer miembro de la audiencia, las risas entre dientes han desembocado en risa general. El gesto parece un tanto preparado, puesto que lo es. Aún, cuando llega el momento de guardar la historia de la impresora láser de Xerox, Stallman lo hace con un gesto ceremonioso de hombre de espectáculo. "El probablemente se lo hizo a la mayoría de las personas que estan aquí Para cuando Stallman ha elegido a un tercer miembro de la audiencia, las tímidas sonrisas han dado paso a una risa general. El gesto parece un tanto preparado, puesto que lo es. Aún, para cuando es la hora de cerrar la hostoria de la impresora láser Xerox, Stallman lo hace con un gesto ceremonioso de hombre del espectáculo. "Probablemente el traicionó a casi todos los que nos encontramos en este salón-- excepto unos pocos, que tal vez, no habían nacido todavía en 1980," Stallman dice, levantando más risas. "[Esto es] porque él prometió rehusarse a cooperar con justamente toda la población del planeta tierra."

Stallman deja que el comentario se asiente por un segundo "El había firmado un acuerdo de no revelar (nondisclosure agreement)," agrega Stallman.

El ascenso de Richard Matthew Stallman de académico frustrado a líder político durante los últimos 20 años habla de muchas cosas. Habla de la naturaleza terca de stallman y de su voluntad prodigiosa. Habla acerca de la visión articulada y los valores del movimiento del software libre que Stallman ayudó a construir. Habla de la alta calidad de los programas de software que Stallman construyó, programas que han cimentado la reputación de Stallman como programador legendario. Habla del momentum creciente de la GPL, una innovación legal, que muchos de los observadores de Stallman ven como su mayor logro.

De manera más importante, habla de la naturaleza cambiante del poder político en un mundo crecientemente obligado por gratitud ante la tecnología de los computadores y los programas de software que potencian esta tecnología.

Quizá por esto, aún en un tiempo en que gran parte de las estrellas de la alta tecnología están declinando, la estrella de Stallman crece. Puesto que el lanzamiento del Proyecto GNU en 1984,[6]

Stallman ha sido a ratos ignorado, satirizado, calumniado, y atacado --tanto desde adentro como desde afuera del movimiento de software libre. A través de todo este proceso, el proyecto GNU ha logrado marcar sus propios hitos, aún con unos pocos notables retrasos, y ha logrado mantener su relevancia en un mercado de software varios ordenes de magnitud más complejo que aquel al cual entraron 18 años antes. Así también ha pasado con la ideología del software libre, una ideología que ha sido meticulosamente cuidada y mantenida por el mismo Stallman.

Para entender las razones detrás de esta situación, ayuda el examinar a Richard Stallman tanto en sus propias palabras como en las palabras de la gente que ha colaborado y batallado al lado de él a lo largo del camino. El esbozo del carácter de Richard Stallman no es complicado. Si cualquier persona ejemplifica el antiguo adagio "lo que usted ve es lo que usted obtiene," ese es Stallman.

"Pienso que si ustde quiere entender a Richard Stallman el ser humano, ustde necesita ver todas las partes como un todo consistente," advierte Eben Moglen, consejero legal de la Fundación de Software Libre y profesor de leyes del Colegio de Leyes de la Universidad de Columbia. "Todas esas excentridades personales que mucha gente ve como obstaculos para llegar a conocer al verdadero Stallman *son* Stallman: el fuerte sentido de frustración personal de Richard, su enorme sentido de compromiso con los principios éticos, su poca capacidad para tranzar, especialmente en aquellos asuntos que considera fundamentales. Estas son las verdaderas razones por las cuales Richard hizo lo que hizo."

Explicar como una jornada que empezó con una impresora láser habría de desembocar eventualmente en una confrontación *sparring* con la más rica corporación del mundo no resulta ser una fácil tarea. Requiere un examen atento de las fuerzas que han hecho de la propiedad del software algo tan importante en la sociedad de hoy en día. También requiere de un examen cuidadoso de un hombre que, como muchos otros líderes políticos antes que él, comprenda la maleabilidad de la memoria humana. Requiere de una habilidad para interpretar los mitos y las palabras-códigos cargadas políticamente que se han construido alrededor de Stallman a través del tiempo. Finalmente, require de la comprensión del genio de Stallman como programador y sus fracasos y éxitos al transmitir tal genio a otras empresas.

Cuando llega el momento de ofrecer su propio resumen de la jornada, Stallman testifica acerca de la fusión de personalidad y principios observadas por Moglen. "La terquedad es mi sello personal ," el dice. "Mucha gente que intenta hacer cualquier cosa que entrañe cierta gran dificultad eventualmente se desalienta y se rinde. Yo nunca me rindo."

El también le da crédito al ciego azar. Si no hubiera sido por ese incidente con la impresora láser Xerox, si no hubiera sido por los conflictos personales y políticos que clausuraron su carrera como empleado del MIT, si no hubiera sido por la media docena de factores que se presentaron de manera sincronizada, Stallman encuentra que hubiera sido muy fácil que su vida hubiera cogido por otro camino diferente. Habiéndose dicho esto, Stallman agradece a las fuerzas y circustancias que lo pusieron en la posición de hacer la diferencia.

"Tenía justamente las destrezas requeridas," dice Stallman, recapitulando su decision de lanzar el Proyecto GNU ante la audiencia. "Nadie más había que yo, así que me sentí como, ‘He sido elegido. He de trabajar en esto. Si no soy yo, quién entonces?"

Notas

[1] De hecho, los poderes de la GPL no son tan potentes. De acuerdo con la sección 10 de la Licencia Pública General GNU, Version 2 (1991), la naturaleza viral de la licencia depende fuertemente de la disposición de la Fundación de Software Libre de ver tal programa como un trabajo derivado, para no mencionar la licencia existente que la GPL reemplazaría. Si usted desea incorporar partes del programa dentro de otros programas libres cuyas condiciones de distribución son diferentes, escriba al autor pidiéndole permiso. Para software de la Fundación de Software Libre, escriba a ésta; algunas veces hacemos excepciones para esto. Nuestra decisión estará guiada por los dos objetivos de preservar el estatus de libertad de todos los derivados de nuestro software libre y el de promover el que se comparta y se reuse del software en general. "El comparar algo a un virus es muy severo," dice Stallman "Una planta de reproducción vegetativa es una comparación más exacta; esta continua reproduciéndose si ustde activamente corta un pié." Para más información acerca de la Licencia Pública General GNU, visite http://www.gnu.org/copyleft/gpl.html.

[2] See Shubha Ghosh, "Revealing the Microsoft Windows Source Code," [Gigalaw.com] (January, 2000). http://www.gigalaw.com/articles/ghosh-2000-01-p1.html

[3] Las aplicaciones matadoras no tienen que ser propietarias. Testigo de esto, por supuetso, el visor Mosaic, un programa cuyos derechos de copia permiten derivados comerciales con ciertas restricciones. Aún, creo que el lector entiende el punto: el mercado del software es como una lotería. Mientras mayor es la ganancia potencial más personas estan dispuestas a participar. Para un buen resumen del fenómeno de las aplicaciones matadoras, vea Philip Ben-David, "Whatever Happened to the ‘Killer App’?" (Qué sucedió con las aplicaciones matadoras?) [e-Commerce News] (December 7, 2000). http://www.ecommercetimes.com/perl/story/5893.html

[4] Vea Craig Mundie, "The Commercial Software Model," (El Modelo de Software Comercial) senior vicepresidente, Microsoft Corp. Citado de un transcrito en línea del discurso de Mundie ante la Escuela Stern de Negocios de la Universidad de Nueva York, el 3 de mayo de 2001. http://www.microsoft.com/presspass/exec/craig/05-03sharedsource.asp

[5] Vea Craig Mundie, "The Commercial Software Model," (El Modelo de Software Comercial) senior vicepresidente, Microsoft Corp. Citado de un transcrito en línea del discurso de Mundie de Mayo 3 de 2001 ante la Escuela Stern de Negocios de la Universidad de Nueva York. http://www.microsoft.com/presspass/exec/craig/05-03sharedsource.asp

[6] El acrónimo GNU significa "GNU no es Unix." En otra porción de su discurso en la Universidad de Nueva York de mayo 29, 2001, Stallman recapituló el origen del acrónimo:

> Nosotros los hackers siempre buscamos un nombre que sea bonito o picante para un programa, puesto que el nombrar un programa es la mitad del gozo de escribir un programa. Nosotros tenemos también la tradición de los acrónimos recursivos, para expresar el hecho que el programa que estamos escribiendo es similar a algún otro programa existente… Estuve buscando por un acrónimo recursivo para Algo No Es UNIX. Y ensayé todas las 26 letras y descubrí que con ninguna de ellas se formaba una palabra. Decidí hacer de ela una contracción. De esta manera tendría un acrónimo de 3 letras, para Algo No es UNIX. Ensayé diferentes letras y me crucé con la palabra "GNU". Esa era. Aun siendo un hincha de los juegos de palabras, Stallman recomienda que los usuarios de software pronuncien la "g" al comienzo del acrónimo (en inglés i.e., "gah-new"). Esto no sólo evita la confusión con la palabra "gnu", el nombre del mamífero Africano, *Connochaetes gnou*, también se evita la confusión con el adjetivo inglés "new". "Hemos estado trabajando en él por los últimos 17 años así que él no es exactamente nuevo," dice Stallman.

Fuente: notas del autor y transcrito en línea de "Free Software: Freedomand Cooperation," charla de Richard Stallman de mayo 29, 2001, en la Universidad de Nueva York. http://www.gnu.org/events/rms-nyu-2001-transcript.txt

---

Capítulo 3. Un retrato del hacker adolescente

La madre de Richard Stallman, Alicia Lippman, todavía recuerda el momento en que notó que su hijo tenia un don especial.

"Creo que fue cuando él tenía ocho años," Lippman recuerda.

El año era 1961, y Lippman, una madre soltera recientemente divorciada, estaba pasando el tiempo una tarde de un fin de semana, dentro del pequeño apartamento familiar del Upper West Side de Manhattan. Hojeando una copia del Scientific American, Lippman llegó a su sección favorita, la columna escrita por Martin Gardner intitulada "Juegos Matemáticos." Lippman, una profesora sustituta de arte, siempre disfrutó la columna de Gardner debido a los retos mentales que ella proveía. Con su hijo ya concentrado en un libro en el sofa contiguo, Lippman decidió tratar de resolver el rompecabezas de la semana.

"Yo no era la mejor en lo que se refería a la resolución de problemas," ella admite. "Pero como artista, encontré que ellos me ayudaban a traspasar barreras conceptuales."

Lippman dice que su intento de resolver el problema encontró inmediatamente una barrera insoslayable. Cercana a arrojar la revista con disgusto, Lippman se vio sorprendida por un suave tirón a la manga de su camisa.

"Era Richard," ella recuerda, "El quería saber si yo necesitaba ayuda."

Oscilando entre el rompecabezas y su hijo, Lippman cuenta que inicialmente ella observó con escepticismo el ofrecimiento de su hijo. "Le pregunté a Richard si él había leído la revista," ella relata. "Me dijo que si, y aún más él ya había resuelto el rompecabezas. A continuación él comenzó a explicarme como resolverlo."

Escuchando la lógica de la forma en que su hijo atacó el problema, el escepticismo de Lippman, rápidamente desembocó en incredulidad. "Quiero decir, siempre supe que era un niño brillante," ella cuenta, "pero esta era la primera vez en que notaba algo que le sugiría; lo avanzado que en realidad estaba."

Treinta años después del hecho, Lippman precisa su memoria con una risa. "Para decir verdad, no creo que nunca habría yo de resolver el rompecabezas," ella relata. "Todo lo que recuerdo es haberme sorprendido de que el supiera la respuesta."

Sentada en el comedor de su segundo apartamento de Manhattan- el mismo complejo espacioso de tres habitaciones que ella y su hijo ocuparían después del matrimonio de ella en 1967 con Maurice Lippman, ya fallecido- Alice Lippman exuda una mezcla de orgullo de madre judía y de perplejidad cuando rememora los primeros años de vida de su hijo. El comedor aparador contiguo ofrece una foto de ocho por diez de Stallman ofreciendo una mirada con el ceño fruncido, con barba completa y traje de doctorado. La imagen hace parecer pequeñas las fotos acompañantes de los sobrinos y nietos de Lippman, pero antes de que el visitante pueda sacar conclusiones de esto, Lippman se asegura de balancear su lugar prominente con un comentario irónico.>

"Richard insistió en que la tuviera después de su doctorado honorario de la Universidad de Glasgow," cuenta Lippman. "El me dijo, ‘Adivina que, mamá? Es la primera ceremonia de graduación a la que he asistido.’"[1]

Tales comentarios reflejan el sentido de humor que se desarrolla cuando se educa a un niño prodigio. No se equivoquen, por cada historia que Lippman escucha o lee acerca de la testarudez y del comportamiento inusual de su hijo, ella puede replicar con al menos una docena de historias similares.

"El solía ser tan conservador," ella cuenta, lanzando sus brazos con una mueca de exasperación. "Solíamos tener las peores discusiones justo aquí en este comedor. Yo hice parte del primer grupo de maestros de escuelas públicas locales que lucharon para formar un sindicato, y Richard estaba muy furioso conmigo. El veía a los sindicatos como corruptos. El, también, se oponía a la seguridad social. El pensaba que la gente podía hacer mucho más dinero invirtiendo en sí mismos. Quien iría a imaginar, que en menos de diez años se habría de volver tan idealista? Todo lo que recuerdo es a su hermana menor venir a mi diciendo, ‘Qué es lo que el será cuando grande? Un fascista?’"

Como madre soltera por cerca de una década -ella y el padre de Richard, Daniel Stallman, que se casaron en 1948, se divorciaron en 1958, y después compartieron la custodia de su hijo- Lippman puede atestiguar acerca de la aversión de su hijo por la autoridad. Ella también puede atestiguar acerca del deseo de conocimiento de su hijo. Fue cuando estas dos fuerzas se conjugaron, Lippman dice, que ella y su hijo experimentaron sus más grandes batallas.

"Fue como si el nunca hubiera querido comer," cuenta Lippman, recordando el patrón de comportamiento que estableció alrededor de los ocho años de edad y que nunca abandonó sino hasta la graduación de bachillerato de 1970 de su hijo. "Lo llamé a comer, y el nunca me oyó. Hube de llamarlo 9 o 10 veces tan sólo para obtener su atención. El estaba totalmente inmerso."

Stallman, por su parte, recuerda cosas de una manera similar, aunque con un giro político.

"Me encantaba leer," el dice. "Cuando quería leer, y mi madre me llamaba a la cocina a comer e ir a dormir, no iba a escucharla. No veía ninguna razón por la cual no habría de seguir leyendo. Ninguna razón para que ella me dijera lo que tenía que hacer, punto. Esencialmente, las ideas sobre las cuales habia leído, tales como democracia y libertad individual, las aplicaba a mí mismo. No veía ninguna razón por la cual los niños habrían de ser excluidos de estos principios."

La creencia en la libertad individual por encima de la autoridad arbitraria se extendió al colegio, también. Con dos años por delante de sus compañeros de clase para cuando tenía 11 años de edad, Stallman sufrió todas las frustraciones usuales en los niños prodigio en las escuelas publicas. No fue mucho después del incidente con el rompecabezas que su madre asistió a la primera de la que habría de ser una larga cadena de charlas entre padres y maestros.

"El se negaba absolutamente a escribir textos," cuenta Lippman, recordando una de sus primeras controversias. "Pienso que el último texto que él escribió antes de su último año de bachillerato fue un ensayo acerca de la historia del sistema numérico occidental en para un profesor de cuarto de primaria."

Con un don para todo aquello que requiriese pensamiento analítico, Stallman se inclinó por las matemáticas y las ciencias a expensas del resto de materias. Lo que algunos profesores vieron como una mente resuelta, sin embargo, Lippman vió como impaciencia. Las matemáticas y la ciencia simplemente ofrecían demasiadas oportunidades para aprender, especialmente en comparación con temas y búsquedas para las cuales su hijo parecía menos naturalmente inclinado. Alrededor de los 10 u 11, cuando los niños en el curso de Stallman comenzaron a jugar regularmente de fútbol, ella recuerda

cuando su hijo regresó a casa envuelto en ira. "El quería tanto jugar, pero simplemente no tenia las destrezas de coordinación requeridas," Lippman recuerda. "Eso lo puso muy furioso."

La rabia eventualmente llevó a su hijo a enfocarse aún más en matemáticas y ciencias. Aun en el reino de la ciencia, sin embargo, la impaciencia de su hijo podría llegar a ser problemática. Estudiando detenidamente los libros de texto de cálculo para cuando contaba siete años de edad, Stallman veia poca necesidad en callar su su discurso ante los adultos. En algún momento, durante sus años de colegio, Lippman contrató un estudiante de la vecina, Universidad de Columbia para que hiciera de hermano mayor de su hijo. El estudiante abandonó el apartamento después de su primera sesión y nunca regresó.

Otra anécdota favorita de la madre data de comienzos de los años 1960, poco después del incidente con el rompecabezas. Cuando tenía alrededor de siete años, dos años depués del divorcio y la relocalización de Queens, Richard tomó el pasatiempo de los modelos de lanzamiento de cohetes en el vecino Parque de Riverside Drive. Lo que comenzó como una diversión inofesiva pronto tomó un cariz de suma seriedad en la medida que su hijo comenzó a registrar datos de cada lanzamiento. Igual que el interés en juegos matemáticos, la actividad atrajo poca atanción hasta que un día, justo antes de un gran lanzamiento de cohetes de la NASA, Lippman inquirió a Richard para ver si estaba interesado en observar.

"El estaba echando humo," Lippman cuenta. "Todo lo que pudo decirme fue, ‘Pero si no estoy listo para publicar todavía.’ Aparentemente el tenía algo que verdaderamente quería mostrarle a la NASA."

Tales anécdotas ofrecen una evidencia temprana de la intensidad que se convertiría en la marca distintiva de Stallman a lo largo de su vida. Cuando otros niños se sentaban a la mesa, Stallman permanecía en su cuarto a leer. Cuando otros niños jugaban a ser Johnny Unitas, Stallman jugaba a ser Werner von Braun. "Yo era extraño," Stallman dice, recordando sus años tempranos sucintamente en una entrevista de 1999. "Después de cierta edad, los únicos amigos que yo tenía eran profesores." [2]

Auncuando esto significara más comparecencias por disciplina en el colegio, Lippman decidió ser indulgente con la pasión de su hijo. Cuando tenía 12 años de edad, Richard atendía a los campamentos de ciencia durante el verano y al colegio privado durante el año escolar. Cuando un profesor le recomendó a ella que lo enrolara en el Programa de Honores de Ciencia de Columbia, un programa post-Sputnik diseñado para estudiantes talentosos de los cursos medios y altos de colegio en New York City, Stallman aceptó aumentar sus actividades extracurriculares y pronto se encontró viajando a la parte alta de la ciudad para llegar al campus de la Universidad de Columbia todos los sábados.

Dan Chess, un compañero de clase en el Programa de Honores de Ciencias de Columbia, recuerda a Richard Stallman pareciéndole un tanto extraño aún entre estudiantes que compartían un deseo similar por matemáticas y ciencias. "Todos eramos geeks y nerds, pero él estaba inusualmente desadaptado," recuerda Chess, ahora un profesor de matemáticas en el Hunter College. "El también era listo como el demonio. He conocido mucha gente lista, pero creo que él era la persona más lista que jamás he conocido."

Seth Breidbart, un compañero y alumno del Programa de Honores de Ciencias de Columbia, ofrece un testomonio reforzador. Un programador de computadores que se ha mantenido en contacto con Richard Stallman gracias a una pasión compartida por la ciencia ficción y las convenciones de ciencia ficción, le recuerda a un Stallman de 15 años *buzz-cut-wearing* tan "intimidante", especialmente para un compañero de 15 años de edad.

"Es difícil de describir," Breidbart dice. "No era como si el fuera inabordable. El era simplemente demasiado intenso. [El era] muy conocedor pero también muy testarudo de ciertas formas."

Tales descripciones invitan a la especulación: los adjetivos cargados de juicio tales como "intenso" y "testarudo" son simplemente una forma de describir rasgos que hoy en día podrían ser categorizados como un desorden juvenil de comportamiento? Un diciembre del 2001, un artículo de la revista [Wired]intitulado "El Síndrome Geek" dibuja el retrato de varios niños científicamente dotados diagnosticados con autismo *high-functioning* o Síndrome de Asperger. De varias formas, los recuerdos paternos registrados en el artículo de [Wired] article are eerily are eerily similar to the ones offered by Lippman. Even Stallman has indulged in psychiatric revisionism from time to time. During a 2000 profile for the [Toronto Star], Stallman described himself to an interviewer as "borderline autistic,"[3] a description that goes a long way toward explaining a lifelong tendency toward social and emotional isolation and the equally lifelong effort to overcome it.

Such speculation benefits from the fast and loose nature of most so-called "behavioral disorders" nowadays, of course. As Steve Silberman, author of "The Geek Syndrome," notes, American psychiatrists have only recently come to accept Asperger Syndrome as a valid umbrella term covering a wide set of behavioral traits. The traits range from poor motor skills and poor socialization to high intelligence and an almost obsessive affinity for numbers, computers, and ordered systems.[4] Reflecting on the broad nature of this umbrella, Stallman says its possible that, if born 40 years later, he might have merited just such a diagnosis. Then again, so would many of his computer-world colleagues.

"It’s possible I could have had something like that," he says. "On the other hand, one of the aspects of that syndrome is difficulty following rhythms. I can dance. In fact, I love following the most complicated rhythms. It’s not clear cut enough to know."

Chess, for one, rejects such attempts at back-diagnosis. "I never thought of him [as] having that sort of thing," he says. "He was just very unsocialized, but then, we all were."

Such descriptions give rise to speculation: are judgment-laden adjectives like "intense" and "hardheaded" simply a way to describe traits that today might be categorized under juvenile behavioral disorder? A December, 2001, [Wired] magazine article titled "The Geek Syndrome" paints the portrait of several scientifically gifted children diagnosed with high-functioning autism or Asperger Syndrome. In many ways, the parental recollections recorded in the [Wired] article Lippman, on the other hand, entertains the possibility. She recalls a few stories from her son’s infancy, however, that provide fodder for speculation. A prominent symptom of autism is an oversensitivity to noises and colors, and Lippman recalls two anecdotes that stand out in this regard. "When Richard was an infant, we’d take him to the beach," she says. "He would start screaming two or three blocks before we reached the surf. It wasn’t until the third time that we figured out what was going on: the sound of the surf was hurting his ears." She also recalls a similar screaming reaction in relation to color: "My mother had bright red hair, and every time she’d stoop down to pick him up, he’d let out a wail."

In recent years, Lippman says she has taken to reading books about autism and believes that such episodes were more than coincidental. "I do feel that Richard had some of the qualities of an autistic child," she says. "I regret that so little was known about autism back then."

Over time, however, Lippman says her son learned to adjust. By age seven, she says, her son had become fond of standing at the front window of subway trains, mapping out and memorizing the labyrinthian system of railroad tracks underneath the city. It was a hobby that relied on an ability to accommodate the loud noises that accompanied each train ride. "Only the initial noise seemed to bother him," says Lippman. "It was as if he got shocked by the sound but his nerves learned how to

make the adjustment."

Para la mayoría de las veces, Lippman recuerda a su hijo exhibiendo la exitación, energía y destrezas sociales de cualquier niño normal. No fue sino después de una serie de eventos traumáticos que destruyeron el hogar de Stallman, ella dice, que su hijo se volvió introvertido y emocionalmente distante.

El primer evento traumático fue el divorcio de Alice y Daniel Stallman, el padre de Richard. Aun cuando Lippman dice que tanto ella como su ex-esposo trataron de preparar a su hijo para el golpe, ella dice que el golpe fue devastador sin embargo. "El hizo como si no hubiera prestado atencióna cuando nosotros le contamos lo que estaba sucediendo," Lippman recuerda. "Pero la realidad le golpeó en la cara cuando él y yo nos trasteamos a un nuevo apartamento. La primera cosa que él dijo fue, ‘Dónde están los muebles de papa?’"

Durante la siguiente década, Stallman habría de emplear sus días de entre semana en el aprtamento de su madre en Manhattan y sus fines de semana en el hogar de su padre en Queens. El ir y venir le dió la oportunidad de estudiar el par de estilos contrastantes de ser padres que, hasta este día, deja a Stallman firmemente opuesto a la idea de criar un niño el mismo. Hablando de su padre, un veterano de la segunda guerra mundial que falleció a inicios del 2001, Stallman balancea respeto con rabia. Por un lado, hay un hombre cuyo compromiso moral lo llevó a aprender francés con el fin de poder ser mas útil para los aliados cuando ellos finalmente vinieron. Por el otro lado, ahí estaba un padre que siempre supo como ingeniarse una humillación para lograr un efecto cruel. [5]

"Mi padre tenía un temperamento horrible," Stallman dice. "El nunca gritó, pero él siempre encontró alguna forma de criticarlo a uno de una manera fría, y diseñado para abatirlo a uno."

En lo que respecta a la vida en el apartamento de su madre, Stallman es menos equívoco. "Eso era guerra," el dice. "Yo solía decir en mi miseria, ‘Quiero irme a casa,’ significando aqule lugar inexistente que yo nunca tendré."

Durante los primeros años después del divorcio, Stallman encontró la tranquilidad que lo eludió en la casa de sus abuelos paternos. Entonces, cuando estaba alrededor de lo 10 años de edad, sus abuelos murieron en una corta sucesión. Para Stallman, la pérdida fue devastadora. "Yo solía ir y visitarlos y me sentía en un ambiente gentil y amoroso," Stallman recuerda. "Este era el único lugar que yo habría de encontrar, hasta cuando entre al college."

Lippman lista la muerte de los abuelos paternos de Richard como el segundo evento traumático. "Esto realmente lo indispuso," ella dice. El era muy cercano a ambos abuelos. Antes de que ellos murieran, él era bastante extrovertido, casi del tipo de líder de la manada cuando estaba con los otros chicos. Después de que ellos murieran, el se volvió mucho más retraido emocionalmente."

Desde la perspectiva de Stallman, el retraimiento emocional era meramente un intento por hacerse cargo con la gonía de la adolescencia. Etiquetando a sus años adolescentes como "horror puro," Stallman dice que él usualmente se sintió como una persona sorda en medio de una multitud de rechinantes escuchadores de música.

"Usualmente tuve el sentimiento que yo nunca pude comprender aquello de cual hablaban las demás personas," dice Stallman, rememorando la burbuja emocional que lo aisló del resto del mudo adulto y adolescente. "Podía entender las palabras, pero algo subyacía en las conversaciones que yo no podía entender. Yo no podía comprender porque la gente estaba interesada en las cosas que otras personas decían."

Para toda la agonía que esto produjo, la adolescencia habría de tener un efecto alentador en el sentido de individualidad de Stallman. En un momento en que gran parte de sus compañeros de clase estaban dejándose crecer el cabello, Stallman prefirió tenerlo corto. En un tiempo en que todo el mundo adolescente estaba escuchando el rock and roll , Stallman prefirió la música clásica. Siendo un fanático devoto de la ciencia ficción,de la revista [Mad] , y de la televisión nocturna, Stallman cultivó una personalidad distintivamente arrolladora que llevó a un límite la incomprensión de sus padres y de sus camaradas por igual.

"Oh, los juegos de palabras," dice Lippman, todavía exasperada por el recuerdo de la personalidad adolescente de su hijo. "No había ninguna cosa que usted pudiera decir en la mesa del comedor que él no te fuera a responder con un juego de palabras."

Fuera del hogar, Stallman guardaba estos chistes para contarselos exclusivamente a los adultos, los cuales tendían a ser indulgentes con su naturaleza virtuosa. Uno de los primeros fue el consejero del campamento de verano que le entregó a Stallman una impresión del manual del computador IBM 7094 cuando él tenía 12 años de edad. Para un preadolescente fascinado con los números y las ciencias, el regalo era como enviado de Dios.[6] Para finales del verano, Stallman estaba escribiendo programas según las especificaciones internas del 7094 internal specifications, anxiosamente anticipando la obtención de una oportunidad de ensayarlos en una máquina real.

Con el primer computador personal todavía a una decada de distancia, Stallman estaría forzado a esperar algunos años antes de obtener acceso a su primer computador. Su primera oportunidad finalmente llegó durante su penúltimo año de bachillerato. Contratado en el Centro Científico IBM de Nueva York, un instituto de investigación, ahora difunto, en el centro comercial de Manhanttan, Stallman empleó el verano después de la graduación de bachiller escribiendo su primer programa, un pre-procesador para el 7094 escrito en el lenguaje de programación PL/I. "Yo primero lo escribí en PL/I, entonces comencé de nuevo en lenguaje assembler cuando el programa en PL/Iestaba demasiado grande para que cupiera en el computador," é recuerda.

Después de ese trabajo en el Centro Científico de IBM, Stallman obtuvo la posición de asistente del laboratorio en el departamento de biología en la Universidad Rockefeller. Aun cuando el ya se estaba encaminando hacia una carrera en matemáticas o física, la mente analítica de Stallman impresionó al director del laboratorio lo suficiente para que años después de que Stallman hubiera partido del colegio, Lippman recibió una inesperada llamada telefónica. "era el profesor del Rockefeller," Lippman dice. "El quería saber cómo le estaba yendo a Richard. El se sorprendió de saber que él estuviera trabajando con computadores. El siempre pensó que Richard tendría un gran futuro adelante como biólogo."

Las destrezas analíticas de Stallman, también impresionaron a los miembros de la facultad en Columbia, aún en el momento en que Stallman mismo se convirtió en blanco de la ira de ellos. "Típicamente una o dos veces por hora [Stallman] habría de descubrir algún error en la conferencia," dice Breidbart. "Y él no era nada tímido al momento de hacerles saber su error a los profesores de manera inmediata. Esto le significó mucho respeto pero no mucha popularidad."

Oyendo la anécdota relatada por Breidbart, este evoca una mirada burlona de Stallman. "Yo era un poco un patán a ratos," el admite. "Pero encontré unos espíritus afines entre los profesores, porque a ellos, también, les gustaba aprender. A los muchachos, para la mayor parte de las veces no les gustaba. Al menos no de la misma forma."

No obstante, pasando el tiempo con los muchachos avanzados durante los sábados, alentó a Stallman a pensar más acerca de los méritos de una socialización mejorada. Con el tiempo de la universidad acercandose, Stallman, como muchos otros de los que estaban en el Programa de Honores en Ciencias en Columbia, había estrechado la lista de sus universidades deseadas a dos escogencias: Harvard y MIT. Sabiendo del deseo de su hijo de trasladarse a la Liga Ivy, Lippman empezó a preocuparse. Siendo él un estudiante de bachillerato de 15 años de edad, Stallman todavía tenía encuentros con los profesores y administradores. Tan sólo un año antes, el había sacado uniformemente Aes en Historia de Estados Unidos, Química, Francés y Algebra, pero una notoria F en Inglés reflejaba el boicott que llevaba a cabo contra las tareas escritas. Tales pifias podría retratar una conocida broma en el MIT, pero en Harvard, ellas eran una bandera roja.

Durante el penúltimo año de su hijo, Lippman dice que ella programó una cita con un terapeuta. El terapeuta expresó instantáneamente una preocupación acerca de la negativa de Stallman a escribir textos y acerca de las reyertas con los profesores. Su hijo ciertamente tenía los requisitos intelectuales para tener éxito en Harvard, pero tendría él la paciencia para sentarse a través de las clases del colegio que requería un *term paper*? El terapeuta sugirió un ensayo. Si Stallman lograba permanecer todo un año en las escuelas públicas de la ciudad de Nueva York, incluyendo una clase de inglés que requiriese la escritura de textos, él probablemente podría sobrevivir a Harvard. Siguiendo la finalización de su penúltimo año, Stallman prontamente se enroló en el colegio de verano en el Louis D. Brandeis High School, una escuela pública localizada en la Calle 84, y comenzo haciendo las clases obligatorias de arte que el había evitado anteriormente durante su carrera de bachillerato.

Para el otoño, Stallman estaba de nuevo dentro de la población de estudiantes de bachillerato de la ciudad de Nueva York que seguía la corriente. No era fácil sentarse en clases que parecían remediales en comparación con sus estudios de los sábados en Columbia, pero Lippman recuerda orgullosamente la habilidad de su hijo para conservar las reglas.

"El fue forzado, hasta un cierto punto a hacer reverencias, pero el lo logró," Lippman dice. "Yo fui llamada una sola vez, lo cual resultaba casi un milagro. Era el profesor de cálculo quejándose de que Richard siempre acusaba al profesor de usar una falsa prueba. Yo le dije, ‘Bien, esta él en lo correcto?’ El profesor dijo, ‘Si, pero yo no puedo decírselo a la clase. Ellos no lo entenderían.’"

Para finales de su primer semestre en Brandeis, las cosas estaban quedando en su lugar. Un 96 en Inglés suprimió gran parte del estigma de los 60 obtenidos 2 años antes. En gran medida, Stallman se escudó con altísimas notas en Historia de los Estados Unidos, Cálculo Avanzado, y Microbiología. El remate fue un perfecto 100 en Física. Aunque todavía siendo un marginado social, Stallman finalizó sus 11 meses en Brandeis como el cuarto mejor estudiante de una clase de 789.

Afuera del aula de clase, Stallman llevó a cabo sus estudioa con una aún mayor diligencia, afanándose entre semana, para cumplir con sus tareas como asistente de laboratorio en la Universidad Rockfeller y evadiendo a los manifestantes contra la Guerra del Vietnam cuando iba camino a la escuela en Columbia. Fue allí, mientras el resto de los estudiantes del Programa de Honores en Ciencias se sentaban a hablar acerca de sus escogencias en la universidad, que Stallman finalmente tomó un momento para participar en la sesión preuniversitaria.

Breidbart recuerda, "Muchos de los estudiantes iban a Harvard o a MIT, naturalmente, pero también había algunos pocos que iban a otras universidades de la Liga Ivy. En la medida que la conversación se movía a través del cuarto, se hacia aparente que Richard no había dicho nada todavía. No sé quien fue pero alguien tuvo el valor de preguntarle acerca de lo que tenía planeado hacer."

Treinta años después, Breidbart recuerda el momento claramente. Tan pronto como Stallman soltó la noticia de que él también iba a asistir a la Universidad de Harvard ese otoño, un extraño silencio llenó el salón. Casi como en una mueca las esquinas de la boca de Stallman se levantaron haciendo una sonrisa de satisfacción personal.

Breidbart dice, "Era la forma silenciosa de de decir, ‘Es correcto. ustedes todavía no se han desembarazado de mi.’"

Notas

[1] Michael Gross, "Richard Stallman: High School Misfit, Symbol of Free Software, MacArthur-certified Genius" (1999). This interview is one of the most candid Stallman interviews on the record. I recommend it highly. http://www.mgross.com/interviews/stallman1.html

[2] Michael Gross, "Richard Stallman: High School Misfit, Symbol of Free Software, MacArthur-certified Genius" (1999). This interview is one of the most candid Stallman interviews on the record. I recommend it highly. http://www.mgross.com/interviews/stallman1.html

[3] See Judy Steed, [Toronto Star], BUSINESS, (October 9, 2000): C03. His vision of free software and social cooperation stands in stark contrast to the isolated nature of his private life. A Glenn Gould-like eccentric, the Canadian pianist was similarly brilliant, articulate, and lonely. Stallman considers himself afflicted, to some degree, by autism: a condition that, he says, makes it difficult for him to interact with people.

[4] See Steve Silberman, "The Geek Syndrome," [Wired] (December, 2001). http://www.wired.com/wired/archive/9.12/aspergers_pr.html

[5] Regrettably, I did not get a chance to interview Daniel Stallman for this book. During the early research for this book, Stallman informed me that his father suffered from Alzheimer’s. When I resumed research in late 2001, I learned, sadly, that Daniel Stallman had died earlier in the year.

[6] Stallman, an atheist, would probably quibble with this description. Suffice it to say, it was something Stallman welcomed. See previous note 1: "As soon as I heard about computers, I wanted to see one and play with one."

Capítulo 4. Impeach God

Although their relationship was fraught with tension, Richard Stallman would inherit one noteworthy trait from his mother: a passion for progressive politics.

It was an inherited trait that would take several decades to emerge, however. For the first few years of his life, Stallman lived in what he now admits was a "political vacuum."[1] Like most Americans during the Eisenhower age, the Stallman family spent the 50s trying to recapture the normalcy lost during the wartime years of the 1940s.

"Richard's father and I were Democrats but happy enough to leave it at that," says Lippman, recalling the family's years in Queens. "We didn't get involved much in local or national politics."

That all began to change, however, in the late 1950s when Alice divorced Daniel Stallman. The move back to Manhattan represented more than a change of address; it represented a new, independent identity and a jarring loss of tranquility.

"I think my first taste of political activism came when I went to the Queens public library and discovered there was only a single book on divorce in the whole library," recalls Lippman. "It was very controlled by the Catholic church, at least in Elmhurst, where we lived. I think that was the first inkling I had of the forces that quietly control our lives."

Returning to her childhood neighborhood, Manhattan's Upper West Side, Lippman was shocked by the changes that had taken place since her departure to Hunter College a decade and a half before. The skyrocketing demand for postwar housing had turned the neighborhood into a political battleground. On one side stood the pro-development city-hall politicians and businessmen hoping to rebuild many of the neighborhood's blocks to accommodate the growing number of white-collar workers moving into the city. On the other side stood the poor Irish and Puerto Rican tenants who had found an affordable haven in the neighborhood.

At first, Lippman didn't know which side to choose. As a new resident, she felt the need for new housing. As a single mother with minimal income, however, she shared the poorer tenants' concern over the growing number of development projects catering mainly to wealthy residents. Indignant, Lippman began looking for ways to combat the political machine that was attempting to turn her neighborhood into a clone of the Upper East Side.

Lippman says her first visit to the local Democratic party headquarters came in 1958. Looking for a day-care center to take care of her son while she worked, she had been appalled by the conditions encountered at one of the city-owned centers that catered to low-income residents. "All I remember is the stench of rotten milk, the dark hallways, the paucity of supplies. I had been a teacher in private nursery schools. The contrast was so great. We took one look at that room and left. That stirred me up."

The visit to the party headquarters proved disappointing, however. Describing it as "the proverbial smoke-filled room," Lippman says she became aware for the first time that corruption within the party might actually be the reason behind the city's thinly disguised hostility toward poor residents. Instead of going back to the headquarters, Lippman decided to join up with one of the many clubs aimed at

reforming the Democratic party and ousting the last vestiges of the Tammany Hall machine. Dubbed the Woodrow Wilson/FDR Reform Democratic Club, Lippman and her club began showing up at planning and city-council meetings, demanding a greater say.

"Our primary goal was to fight Tammany Hall, Carmine DeSapio and his henchman,"[2] says Lippman. "I was the representative to the city council and was very much involved in creating a viable urban-renewal plan that went beyond simply adding more luxury housing to the neighborhood."

Such involvement would blossom into greater political activity during the 1960s. By 1965, Lippman had become an "outspoken" supporter for political candidates like William Fitts Ryan, a Democratic elected to Congress with the help of reform clubs and one of the first U.S. representatives to speak out against the Vietnam War.

It wasn't long before Lippman, too, was an outspoken opponent of U.S. involvement in Indochina. "I was against the Vietnam war from the time Kennedy sent troops," she says. "I had read the stories by reporters and journalists sent to cover the early stages of the conflict. I really believed their forecast that it would become a quagmire."

Such opposition permeated the Stallman-Lippman household. In 1967, Lippman remarried. Her new husband, Maurice Lippman, a major in the Air National Guard, resigned his commission to demonstrate his opposition to the war. Lippman's stepson, Andrew Lippman, was at MIT and temporarily eligible for a student deferment. Still, the threat of induction should that deferment disappear, as it eventually did, made the risk of U.S. escalation all the more immediate. Finally, there was Richard who, though younger, faced the prospect of choosing between Vietnam or Canada when the war lasted into the 1970s.

"Vietnam was a major issue in our household," says Lippman. "We talked about it constantly: what would we do if the war continued, what steps Richard or his stepbrother would take if they got drafted. We were all opposed to the war and the draft. We really thought it was immoral."

For Stallman, the Vietnam War elicited a complex mixture of emotions: confusion, horror, and, ultimately, a profound sense of political impotence. As a kid who could barely cope in the mild authoritarian universe of private school, Stallman experienced a shiver whenever the thought of Army boot camp presented itself.

"I was devastated by the fear, but I couldn't imagine what to do and didn't have the guts to go demonstrate," recalls Stallman, whose March 18th birthday earned him a dreaded low number in the draft lottery when the federal government finally eliminated college deferments in 1971. "I couldn't envision moving to Canada or Sweden. The idea of getting up by myself and moving somewhere. How could I do that? I didn't know how to live by myself. I wasn't the kind of person who felt confident in approaching things like that."

Stallman says he was both impressed and shamed by the family members who did speak out. Recalling a bumper sticker on his father's car likening the My Lai massacre to similar Nazi atrocities in World War II, he says he was "excited" by his father's gesture of outrage. "I admired him for doing it," Stallman says. "But I didn't imagine that I could do anything. I was afraid that the juggernaut of the draft was going to destroy me."

Although descriptions of his own unwillingness to speak out carry a tinge of nostalgic regret, Stallman says he was ultimately turned off by the tone and direction of the anti-war movement. Like other members of the Science Honors Program, he saw the weekend demonstrations at Columbia as little more than a distracting spectacle. [3] Ultimately, Stallman says, the irrational forces driving the

anti-war movement became indistinguishable from the irrational forces driving the rest of youth culture. Instead of worshiping the Beatles, girls in Stallman's age group were suddenly worshiping firebrands like Abbie Hoffman and Jerry Rubin. To a kid already struggling to comprehend his teenage peers, escapist slogans like "make love not war" had a taunting quality. Not only was it a reminder that Stallman, the short-haired outsider who hated rock 'n' roll, detested drugs, and didn't participate in campus demonstrations, wasn't getting it politically; he wasn't "getting it" sexually either.

"I didn't like the counter culture much," Stallman admits. "I didn't like the music. I didn't like the drugs. I was scared of the drugs. I especially didn't like the anti-intellectualism, and I didn't like the prejudice against technology. After all, I loved a computer. And I didn't like the mindless anti-Americanism that I often encountered. There were people whose thinking was so simplistic that if they disapproved of the conduct of the U.S. in the Vietnam War, they had to support the North Vietnamese. They couldn't imagine a more complicated position, I guess."

Such comments alleviate feelings of timidity. They also underline a trait that would become the key to Stallman's own political maturation. For Stallman, political confidence was directly proportionate to personal confidence. By 1970, Stallman had become confident in few things outside the realm of math and science. Nevertheless, confidence in math gave him enough of a foundation to examine the anti-war movement in purely logical terms. In the process of doing so, Stallman had found the logic wanting. Although opposed to the war in Vietnam, Stallman saw no reason to disavow war as a means for defending liberty or correcting injustice. Rather than widen the breach between himself and his peers, however, Stallman elected to keep the analysis to himself.

In 1970, Stallman left behind the nightly dinnertime conversations about politics and the Vietnam War as he departed for Harvard. Looking back, Stallman describes the transition from his mother's Manhattan apartment to life in a Cambridge dorm as an "escape." Peers who watched Stallman make the transition, however, saw little to suggest a liberating experience.

"He seemed pretty miserable for the first while at Harvard," recalls Dan Chess, a classmate in the Science Honors Program who also matriculated at Harvard. "You could tell that human interaction was really difficult for him, and there was no way of avoiding it at Harvard. Harvard was an intensely social kind of place."

To ease the transition, Stallman fell back on his strengths: math and science. Like most members of the Science Honors Program, Stallman breezed through the qualifying exam for Math 55, the legendary "boot camp" class for freshman mathematics "concentrators" at Harvard. Within the class, members of the Science Honors Program formed a durable unit. "We were the math mafia," says Chess with a laugh. "Harvard was nothing, at least compared with the SHP."

To earn the right to boast, however, Stallman, Chess, and the other SHP alumni had to get through Math 55. Promising four years worth of math in two semesters, the course favored only the truly devout. "It was an amazing class," says David Harbater, a former "math mafia" member and now a professor of mathematics at the University of Pennsylvania. "It's probably safe to say there has never been a class for beginning college students that was that intense and that advanced. The phrase I say to people just to get it across is that, among other things, by the second semester we were discussing the differential geometry of Banach manifolds. That's usually when their eyes bug out, because most people don't start talking about Banach manifolds until their second year of graduate school."

Starting with 75 students, the class quickly melted down to 20 by the end of the second semester. Of that 20, says Harbater, "only 10 really knew what they were doing." Of that 10, 8 would go on to become future mathematics professors, 1 would go on to teach physics.

"The other one," emphasizes Harbater, "was Richard Stallman."

Seth Breidbart, a fellow Math 55 classmate, remembers Stallman distinguishing himself from his peers even then.

"He was a stickler in some very strange ways," says Breidbart. "There is a standard technique in math which everybody does wrong. It's an abuse of notation where you have to define a function for something and what you do is you define a function and then you prove that it's well defined. Except the first time he did and presented it, he defined a relation and proved that it's a function. It's the exact same proof, but he used the correct terminology, which no one else did. That's just the way he was."

It was in Math 55 that Richard Stallman began to cultivate a reputation for brilliance. Breidbart agrees, but Chess, whose competitive streak refused to yield, says the realization that Stallman might be the best mathematician in the class didn't set in until the next year. "It was during a class on Real Analysis, which I took with Richard the next year," says Chess, now a math professor at Hunter College. "I actually remember in a proof about complex valued measures that Richard came up with an idea that was basically a metaphor from the calculus of variations. It was the first time I ever saw somebody solve a problem in a brilliantly original way."

Chess makes no bones about it: watching Stallman's solution unfold on the chalkboard was a devastating blow. As a kid who'd always taken pride in being the smartest mathematician the room, it was like catching a glimpse of his own mortality. Years later, as Chess slowly came to accept the professional rank of a good-but-not-great mathematician, he had Stallman's sophomore-year proof to look back on as a taunting early indicator.

"That's the thing about mathematics," says Chess. "You don't have to be a first-rank mathematician to recognize first-rate mathematical talent. I could tell I was up there, but I could also tell I wasn't at the first rank. If Richard had chosen to be a mathematician, he would have been a first-rank mathematician."

For Stallman, success in the classroom was balanced by the same lack of success in the social arena. Even as other members of the math mafia gathered to take on the Math 55 problem sets, Stallman preferred to work alone. The same went for living arrangements. On the housing application for Harvard, Stallman clearly spelled out his preferences. "I said I preferred an invisible, inaudible, intangible roommate," he says. In a rare stroke of bureaucratic foresight, Harvard's housing office accepted the request, giving Stallman a one-room single for his freshman year.

Breidbart, the only math-mafia member to share a dorm with Stallman that freshman year, says Stallman slowly but surely learned how to interact with other students. He recalls how other dorm mates, impressed by Stallman's logical acumen, began welcoming his input whenever an intellectual debate broke out in the dining club or dorm commons.

"We had the usual bull sessions about solving the world's problems or what would be the result of something," recalls Breidbart. "Say somebody discovers an immortality serum. What do you do? What are the political results? If you give it to everybody, the world gets overcrowded and everybody dies. If you limit it, if you say everyone who's alive now can have it but their children can't, then you end up with an underclass of people without it. Richard was just better able than most to see the unforeseen circumstances of any decision."

Stallman remembers the discussions vividly. "I was always in favor of immortality," he says. "I was shocked that most people regarded immortality as a bad thing. How else would we be able to see what the world is like 200 years from now?"

Although a first-rank mathematician and first-rate debater, Stallman shied away from clear-cut competitive events that might have sealed his brilliant reputation. Near the end of freshman year at Harvard, Breidbart recalls how Stallman conspicuously ducked the Putnam exam, a prestigious test open to math students throughout the U.S. and Canada. In addition to giving students a chance to measure their knowledge in relation to their peers, the Putnam served as a chief recruiting tool for academic math departments. According to campus legend, the top scorer automatically qualified for a graduate fellowship at any school of his choice, including Harvard.

Like Math 55, the Putnam was a brutal test of merit. A six-hour exam in two parts, it seemed explicitly designed to separate the wheat from the chaff. Breidbart, a veteran of both the Science Honors Program and Math 55, describes it as easily the most difficult test he ever took. "Just to give you an idea of how difficult it was," says Breidbart, "the top score was a 120, and my score the first year was in the 30s. That score was still good enough to place me 101st in the country."

Surprised that Stallman, the best student in the class, had passed on the test, Breidbart says he and a fellow classmate cornered him in the dining common and demanded an explanation. "He said he was afraid of not doing well," Breidbart recalls.

Breidbart and the friend quickly wrote down a few problems from memory and gave them to Stallman. "He solved all of them," Breidbart says, "leading me to conclude that by not doing well, he either meant coming in second or getting something wrong."

Stallman remembers the episode a bit differently. "I remember that they did bring me the questions and it’s possible that I solved one of them, but I’m pretty sure I didn’t solve them all," he says. Nevertheless, Stallman agrees with Breidbart’s recollection that fear was the primary reason for not taking the test. Despite a demonstrated willingness to point out the intellectual weaknesses of his peers and professors in the classroom, Stallman hated the notion of head-to-head competition.

"It’s the same reason I never liked chess," says Stallman. "Whenever I’d play, I would become so consumed by the fear of making a single mistake that I would start making stupid mistakes very early in the game. The fear became a self-fulfilling prophecy."

Whether such fears ultimately prompted Stallman to shy away from a mathematical career is a moot issue. By the end of his freshman year at Harvard, Stallman had other interests pulling him away from the field. Computer programming, a latent fascination throughout Stallman’s high-school years, was becoming a full-fledged passion. Where other math students sought occasional refuge in art and history classes, Stallman sought it in the computer-science laboratory.

For Stallman, the first taste of real computer programming at the IBM New York Scientific Center had triggered a desire to learn more. "Toward the end of my first year at Harvard school, I started to have enough courage to go visit computer labs and see what they had. I’d ask them if they had extra copies of any manuals that I could read."

Taking the manuals home, Stallman would examine machine specifications, compare them with other machines he already knew, and concoct a trial program, which he would then bring back to the lab along with the borrowed manual. Although some labs balked at the notion of a strange kid coming off the street and working on the lab machinery, most recognized competence when they saw it and let Stallman run the programs he had created.

One day, near the end of freshman year, Stallman heard about a special laboratory near MIT. The laboratory was located on the ninth floor an off-campus building in Tech Square, the newly built facility dedicated to advanced research. According to the rumors, the lab itself was dedicated to the

cutting-edge science of artificial intelligence and boasted the cutting-edge machines and software programs to match.

Intrigued, Stallman decided to pay a visit.

The trip was short, about 2 miles on foot, 10 minutes by train, but as Stallman would soon find out, MIT and Harvard can feel like opposite poles of the same planet. With its maze-like tangle of interconnected office buildings, the Institute's campus offered an aesthetic yin to Harvard's spacious colonial-village yang. The same could be said for the student body, a geeky collection of ex-high school misfits known more for its predilection for pranks than its politically powerful alumni.

The yin-yang relationship extended to the AI Lab as well. Unlike Harvard computer labs, there was no grad-student gatekeeper, no clipboard waiting list for terminal access, no explicit atmosphere of "look but don't touch." Instead, Stallman found only a collection of open terminals and robotic arms, presumably the artifacts of some A.I. experiment.

Although the rumors said anybody could sit down at the terminals, Stallman decided to stick with the original plan. When he encountered a lab employee, he asked if the lab had any spare manuals it could loan to an inquisitive student. "They had some, but a lot of things weren't documented," Stallman recalls. "They were hackers after all."

Stallman left with something even better than a manual: a job. Although he doesn't remember what the first project was, he does remember coming back to the AI Lab the next week, grabbing an open terminal and writing software code.

Looking back, Stallman sees nothing unusual in the AI Lab's willingness to accept an unproven outsider at first glance. "That's the way it was back then," he says. "That's the way it still is now. I'll hire somebody when I meet him if I see he's good. Why wait? Stuffy people who insist on putting bureaucracy into everything really miss the point. If a person is good, he shouldn't have to go through a long, detailed hiring process; he should be sitting at a computer writing code."

To get a taste of "bureaucratic and stuffy," Stallman need only visit the computer labs at Harvard. There, access to the terminals was doled out according to academic rank. As an undergrad, Stallman usually had to sign up or wait until midnight, about the time most professors and grad students finished their daily work assignments. The waiting wasn't difficult, but it was frustrating. Waiting for a public terminal, knowing all the while that a half dozen equally usable machines were sitting idle inside professors' locked offices, seemed the height of illogic. Although Stallman paid the occasional visit to the Harvard computer labs, he preferred the more egalitarian policies of the AI Lab. "It was a breath of fresh air," he says. "At the AI Lab, people seemed more concerned about work than status."

Stallman quickly learned that the AI Lab's first-come, first-served policy owed much to the efforts of a vigilant few. Many were holdovers from the days of Project MAC, the Department of Defense-funded research program that had given birth to the first time-share operating systems. A few were already legends in the computing world. There was Richard Greenblatt, the lab's in-house Lisp expert and author of MacHack, the computer chess program that had once humbled A.I. critic Hubert Dreyfus. There was Gerald Sussman, original author of the robotic block-stacking program HACKER. And there was Bill Gosper, the in-house math whiz already in the midst of an 18-month hacking bender triggered by the philosophical implications of the computer game LIFE. [4]

Members of the tight-knit group called themselves "hackers." Over time, they extended the "hacker" description to Stallman as well. In the process of doing so, they inculcated Stallman in the ethical traditions of the "hacker ethic." To be a hacker meant more than just writing programs, Stallman

learned. It meant writing the best possible programs. It meant sitting at a terminal for 36 hours straight if that's what it took to write the best possible programs. Most importantly, it meant having access to the best possible machines and the most useful information at all times. Hackers spoke openly about changing the world through software, and Stallman learned the instinctual hacker disdain for any obstacle that prevented a hacker from fulfilling this noble cause. Chief among these obstacles were poor software, academic bureaucracy, and selfish behavior.

Stallman also learned the lore, stories of how hackers, when presented with an obstacle, had circumvented it in creative ways. Stallman learned about "lock hacking," the art of breaking into professors' offices to "liberate" sequestered terminals. Unlike their pampered Harvard counterparts, MIT faculty members knew better than to treat the AI Lab's terminal as private property. If a faculty member made the mistake of locking away a terminal for the night, hackers were quick to correct the error. Hackers were equally quick to send a message if the mistake repeated itself. "I was actually shown a cart with a heavy cylinder of metal on it that had been used to break down the door of one professor's office," [5] Stallman says.

Such methods, while lacking in subtlety, served a purpose. Although professors and administrators outnumbered hackers two-to-one inside the AI Lab, the hacker ethic prevailed. Indeed, by the time of Stallman's arrival at the AI Lab, hackers and the AI Lab administration had coevolved into something of a symbiotic relationship. In exchange for fixing the machines and keeping the software up and running, hackers earned the right to work on favorite pet projects. Often, the pet projects revolved around improving the machines and software programs even further. Like teenage hot-rodders, most hackers viewed tinkering with machines as its own form of entertainment.

Nowhere was this tinkering impulse better reflected than in the operating system that powered the lab's central PDP-6 mini-computer. Dubbed ITS, short for the Incompatible Time Sharing system, the operating system incorporated the hacking ethic into its very design. Hackers had built it as a protest to Project MAC's original operating system, the Compatible Time Sharing System, CTSS, and named it accordingly. At the time, hackers felt the CTSS design too restrictive, limiting programmers' power to modify and improve the program's own internal architecture if needed. According to one legend passed down by hackers, the decision to build ITS had political overtones as well. Unlike CTSS, which had been designed for the IBM 7094, ITS was built specifically for the PDP-6. In letting hackers write the systems themselves, AI Lab administrators guaranteed that only hackers would feel comfortable using the PDP-6. In the feudal world of academic research, the gambit worked. Although the PDP-6 was co-owned in conjunction with other departments, A.I. researchers soon had it to themselves.[6]

ITS boasted features most commercial operating systems wouldn't offer for years, features such as multitasking, debugging, and full-screen editing capability. Using it and the PDP-6 as a foundation, the Lab had been able to declare independence from Project MAC shortly before Stallman's arrival. [7]

As an apprentice hacker, Stallman quickly became enamored with ITS. Although forbidding to most newcomers, the program contained many built-in features that provided a lesson in software development to hacker apprentices such as himself.

"ITS had a very elegant internal mechanism for one program to examine another," says Stallman, recalling the program. "You could examine all sorts of status about another program in a very clean, well-specified way."

Using this feature, Stallman was able to watch how programs written by hackers processed instructions as they ran. Another favorite feature would allow the monitoring program to freeze the monitored program's job between instructions. In other operating systems, such a command would have resulted in half-computed gibberish or an automatic systems crash. In ITS, it provided yet another way to monitor the step-by-step performance.

"If you said, 'Stop the job,' it would always be stopped in user mode. It would be stopped between two user-mode instructions, and everything about the job would be consistent for that point," Stallman says. "If you said, 'Resume the job,' it would continue properly. Not only that, but if you were to change the status of the job and then change it back, everything would be consistent. There was no hidden status anywhere."

By the end of 1970, hacking at the AI Lab had become a regular part of Stallman's weekly schedule. From Monday to Thursday, Stallman devoted his waking hours to his Harvard classes. As soon as Friday afternoon arrived, however, he was on the T, heading down to MIT for the weekend. Stallman usually timed his arrival to coincide with the ritual food run. Joining five or six other hackers in their nightly quest for Chinese food, he would jump inside a beat-up car and head across the Harvard Bridge into nearby Boston. For the next two hours, he and his hacker colleagues would discuss everything from ITS to the internal logic of the Chinese language and pictograph system. Following dinner, the group would return to MIT and hack code until dawn.

For the geeky outcast who rarely associated with his high-school peers, it was a heady experience, suddenly hanging out with people who shared the same predilection for computers, science fiction, and Chinese food. "I remember many sunrises seen from a car coming back from Chinatown," Stallman would recall nostalgically, 15 years after the fact in a speech at the Swedish Royal Technical Institute. "It was actually a very beautiful thing to see a sunrise, 'cause that's such a calm time of day. It's a wonderful time of day to get ready to go to bed. It's so nice to walk home with the light just brightening and the birds starting to chirp; you can get a real feeling of gentle satisfaction, of tranquility about the work that you have done that night."[8]

The more Stallman hung out with the hackers, the more he adopted the hacker worldview. Already committed to the notion of personal liberty, Stallman began to infuse his actions with a sense of communal responsibility. When others violated the communal code, Stallman was quick to speak out. Within a year of his first visit, Stallman was the one breaking into locked offices, trying to recover the sequestered terminals that belonged to the lab community as a whole. In true hacker fashion, Stallman also sought to make his own personal contribution to the art of lock hacking. One of the most artful door-opening tricks, commonly attributed to Greenblatt, involved bending a stiff wire into a cane and attaching a loop of tape to the long end. Sliding the wire under the door, a hacker could twist and rotate the wire so that the long end touched the door knob. Provided the adhesive on the tape held, a hacker could open the doorknob with a few sharp twists.

When Stallman tried the trick, he found it good but wanting in a few places. Getting the tape to stick wasn't always easy, and twisting the wire in a way that turned the doorknob was similarly difficult. Stallman remembered that the hallway ceiling possessed tiles that could be slid away. Some hackers, in fact, had used the false ceiling as a way to get around locked doors, an approach that generally covered the perpetrator in fiberglass but got the job done.

Stallman considered an alternative approach. What if, instead of slipping a wire under the door, a hacker slid away one of the panels and stood over the door jamb?

Stallman took it upon himself to try it out. Instead of using a wire, Stallman draped out a long U-shaped loop of magnetic tape, fastening a loop of adhesive tape at the base of the U. Standing over the door jamb, he dangled the tape until it looped under the doorknob. Lifting the tape until the adhesive fastened, he then pulled on the left end of the tape, twisting the doorknob counter-clockwise. Sure enough, the door opened. Stallman had added a new twist to the art of lock hacking.

"Sometimes you had to kick the door after you turned the door knob," says Stallman, recalling the lingering bugginess of the new method. "It took a little bit of balance to pull it off."

Such activities reflected a growing willingness on Stallman's part to speak and act out in defense of political beliefs. The AI Lab's spirit of direct action had proved inspirational enough for Stallman to break out of the timid impotence of his teenage years. Breaking into an office to free a terminal wasn't the same as taking part in a protest march, but it was effective in ways that most protests weren't. It solved the problem at hand.

By the time of his last years at Harvard, Stallman was beginning to apply the whimsical and irreverent lessons of the AI Lab back at school.

"Did he tell you about the snake?" his mother asks at one point during an interview. "He and his dorm mates put a snake up for student election. Apparently it got a considerable number of votes."

Stallman verifies the snake candidacy with a few caveats. The snake was a candidate for election within Currier House, Stallman's dorm, not the campus-wide student council. Stallman does remember the snake attracting a fairly significant number of votes, thanks in large part to the fact that both the snake and its owner both shared the same last name. "People may have voted for it, because they thought they were voting for the owner," Stallman says. "Campaign posters said that the snake was 'slithering for' the office. We also said it was an 'at large' candidate, since it had climbed into the wall through the ventilating unit a few weeks before and nobody knew where it was."

Running a snake for dorm council was just one of several election-related pranks. In a later election, Stallman and his dorm mates nominated the house master's son. "His platform was mandatory retirement at age seven," Stallman recalls. Such pranks paled in comparison to the fake-candidate pranks on the MIT campus, however. One of the most successful fake-candidate pranks was a cat named Woodstock, which actually managed to outdraw most of the human candidates in a campus-wide election. "They never announced how many votes Woodstock got, and they treated those votes as spoiled ballots," Stallman recalls. "But the large number of spoiled ballots in that election suggested that Woodstock had actually won. A couple of years later, Woodstock was suspiciously run over by a car. Nobody knows if the driver was working for the MIT administration." Stallman says he had nothing to do with Woodstock's candidacy, "but I admired it."[9]

At the AI Lab, Stallman's political activities had a sharper-edged tone. During the 1970s, hackers faced the constant challenge of faculty members and administrators pulling an end-run around ITS and its hacker-friendly design. One of the first attempts came in the mid-1970s, as more and more faculty members began calling for a file security system to protect research data. Most other computer labs had installed such systems during late 1960s, but the AI Lab, through the insistence of Stallman and other hackers, remained a security-free zone.

For Stallman, the opposition to security was both ethical and practical. On the ethical side, Stallman pointed out that the entire art of hacking relied on intellectual openness and trust. On the practical side, he pointed to the internal structure of ITS being built to foster this spirit of openness, and any attempt to reverse that design required a major overhaul.

"The hackers who wrote the Incompatible Timesharing System decided that file protection was usually used by a self-styled system manager to get power over everyone else," Stallman would later explain. "They didn't want anyone to be able to get power over them that way, so they didn't implement that kind of a feature. The result was, that whenever something in the system was broken, you could always fix it." [10]

Through such vigilance, hackers managed to keep the AI Lab's machines security-free. Over at the nearby MIT Laboratory for Computer Sciences, however, security-minded faculty members won the day. The LCS installed its first password-based system in 1977. Once again, Stallman took it upon himself to correct what he saw as ethical laxity. Gaining access to the software code that controlled the password system, Stallman implanted a software command that sent out a message to any LCS user who attempted to choose a unique password. If a user entered "starfish," for example, the message came back something like:

> I see you chose the password "starfish." I suggest that you switch to the password "carriage return." It's much easier to type, and also it stands up to the principle that there should be no passwords.[11]

Users who did enter "carriage return"--that is, users who simply pressed the Enter or Return button, entering a blank string instead of a unique password--left their accounts accessible to the world at large. As scary as that might have been for some users, it reinforced the hacker notion that Institute computers, and even Institute computer files, belonged to the public, not private individuals. Stallman, speaking in an interview for the 1984 book [Hackers], proudly noted that one-fifth of the LCS staff accepted this argument and employed the blank-string password. [12]

Stallman's null-string crusade would prove ultimately futile. By the early 1980s, even the AI Lab's machines were sporting password-based security systems. Even so, it represents a major milestone in terms of Stallman's personal and political maturation. To the objective observer familiar with Stallman's later career, it offers a convenient inflection point between the timid teenager afraid to speak out even on issues of life-threatening importance and the adult activist who would soon turn needling and cajoling into a full-time occupation.

In voicing his opposition to computer security, Stallman drew on many of the forces that had shaped his early life: hunger for knowledge, distaste for authority, and frustration over hidden procedures and rules that rendered some people clueless outcasts. He would also draw on the ethical concepts that would shape his adult life: communal responsibility, trust, and the hacker spirit of direct action. Expressed in software-computing terms, the null string represents the 1.0 version of the Richard Stallman political worldview-incomplete in a few places but, for the most part, fully mature.

Looking back, Stallman hesitates to impart too much significance to an event so early in his hacking career. "In that early stage there were a lot of people who shared my feelings," he says. "The large number of people who adopted the null string as their password was a sign that many people agreed that it was the proper thing to do. I was simply inclined to be an activist about it."

Stallman does credit the AI Lab for awakening that activist spirit, however. As a teenager, Stallman had observed political events with little idea as to how a single individual could do or say anything of importance. As a young adult, Stallman was speaking out on matters in which he felt supremely confident, matters such as software design, communal responsibility, and individual freedom. "I joined this community which had a way of life which involved respecting each other's freedom," he says. "It didn't take me long to figure out that that was a good thing. It took me longer to come to the conclusion that this was a moral issue."

Hacking at the AI Lab wasn’t the only activity helping to boost Stallman’s esteem. During the middle of his sophomore year at Harvard, Stallman had joined up with a dance troupe that specialized in folk dances. What began as a simple attempt to meet women and expand his social horizons soon expanded into yet another passion alongside hacking. Dancing in front of audiences dressed in the native garb of a Balkan peasant, Stallman no longer felt like the awkward, uncoordinated 10-year-old whose attempts to play football had ended in frustration. He felt confident, agile, and alive. For a brief moment, he even felt a hint of emotional connection. He soon found being in front of an audience fun, and it wasn’t long thereafter that he began craving the performance side of dancing almost as much as the social side.

Although the dancing and hacking did little to improve Stallman’s social standing, they helped him overcome the feelings of weirdness that had clouded his pre-Harvard life. Instead of lamenting his weird nature, Stallman found ways to celebrate it. In 1977, while attending a science-fiction convention, he came across a woman selling custom-made buttons. Excited, Stallman ordered a button with the words "Impeach God" emblazoned on it.

For Stallman, the "Impeach God" message worked on many levels. An atheist since early childhood, Stallman first saw it as an attempt to set a "second front" in the ongoing debate on religion. "Back then everybody was arguing about God being dead or alive," Stallman recalls. "‘Impeach God’ approached the subject of God from a completely different viewpoint. If God was so powerful as to create the world and yet do nothing to correct the problems in it, why would we ever want to worship such a God? Wouldn’t it be better to put him on trial?"

At the same time, "Impeach God" was a satirical take on America and the American political system. The Watergate scandal of the 1970s affected Stallman deeply. As a child, Stallman had grown up mistrusting authority. Now, as an adult, his mistrust had been solidified by the culture of the AI Lab hacker community. To the hackers, Watergate was merely a Shakespearean rendition of the daily power struggles that made life such a hassle for those without privilege. It was an outsized parable for what happened when people traded liberty and openness for security and convenience.

Buoyed by growing confidence, Stallman wore the button proudly. People curious enough to ask him about it received the same well-prepared spiel. "My name is Jehovah," Stallman would say. "I have a special plan to save the universe, but because of heavenly security reasons I can’t tell you what that plan is. You’re just going to have to put your faith in me, because I see the picture and you don’t. You know I’m good because I told you so. If you don’t believe me, I’ll throw you on my enemies list and throw you in a pit where Infernal Revenue Service will audit your taxes for eternity."

Those who interpreted the spiel as a word-for-word parody of the Watergate hearings only got half the message. For Stallman, the other half of the message was something only his fellow hackers seemed to be hearing. One hundred years after Lord Acton warned about absolute power corrupting absolutely, Americans seemed to have forgotten the first part of Acton’s truism: power, itself, corrupts. Rather than point out the numerous examples of petty corruption, Stallman felt content voicing his outrage toward an entire system that trusted power in the first place.

"I figured why stop with the small fry," says Stallman, recalling the button and its message. "If we went after Nixon, why not going after Mr. Big. The way I see it, any being that has power and abuses it deserves to have that power taken away."

Notas

[1] See Michael Gross, "Richard Stallman: High School Misfit, Symbol of Free Software, MacArthur-certified Genius" (1999).

[2] Carmine DeSapio holds the dubious distinction of being the first Italian-American boss of Tammany Hall, the New York City political machine. For more information on DeSapio and the politics of post-war New York, see John Davenport, "Skinning the Tiger: Carmine DeSapio and the End of the Tammany Era," [New York Affairs] (1975): 3:1.

[3] Chess, another Columbia Science Honors Program alum, describes the protests as "background noise." "We were all political," he says, "but the SHP was imporant. We would never have skipped it for a demonstration."

[4] See Steven Levy, [Hackers] (Penguin USA [paperback], 1984): 144. Levy devotes about five pages to describing Gosper's fascination with LIFE, a math-based software game first created by British mathematician John Conway. I heartily recommend this book as a supplement, perhaps even a prerequisite, to this one.

[5] Gerald Sussman, an MIT faculty member and hacker whose work at the AI Lab predates Stallman's, disputes this memory. According to Sussman, the hackers never broke any doors to retrieve terminals.

[6] I apologize for the whirlwind summary of ITS' genesis, an operating system many hackers still regard as the epitome of the hacker ethos. For more information on the program's political significance, see Simson Garfinkel, [Architects of the Information Society: Thirty-Five Years of the Laboratory for Computer Science at MIT] (MIT Press, 1999).

[7] I apologize for the whirlwind summary of ITS' genesis, an operating system many hackers still regard as the epitome of the hacker ethos. For more information on the program's political significance, see Simson Garfinkel, [Architects of the Information Society: Thirty-Five Years of the Laboratory for Computer Science at MIT] (MIT Press, 1999).

[8] See Richard Stallman, "RMS lecture at KTH (Sweden)," (October 30, 1986). http://www.gnu.org/philosophy/stallman-kth.html

[9] In an email shortly after this book went into its final edit cycle, Stallman says he drew political inspiration from the Harvard campus as well. "In my first year of Harvard, in a Chinese History class, I read the story of the first revolt against the Chin dynasty," he says. "The story is not reliable history, but it was very moving."

[10] See Richard Stallman (1986).

[11] See Steven Levy, [Hackers] (Penguin USA [paperback], 1984): 417. I have modified this quote, which Levy also uses as an excerpt, to illustrate more directly how the program might reveal the false security of the system. Levy uses the placeholder "[such and such]."

[12] See Steven Levy, [Hackers] (Penguin USA [paperback], 1984): 417.

Capítulo 5. Pequeño Charco de Libertad

Pregúntele a cualquiera que haya permanecido más de un minuto en presencia de Richard Stallman y obtendra los mismos comentarios: olvídese del cabello largo. Olvídese del comportamiento extravagante. Lo primero que se nota es la mirada, prolongada y penetrante. Con una sola mirada a los ojos verdes de Stallman, usted sabe que se encuentra ante un verdadero creyente.

Calificar a la mirada de Stallman como intensa es subestimarla. Los ojos de Stallman no solo miran hacía usted; miran a través de usted. Incluso cuando sus propios ojos momentáneamente cambian de dirección, por simple cortesía, los ojos de Stallman permanecen bloqueados, ardiendo en el lado de su cabeza como rayos de fotones gemelos.

Tal vez sea por eso que muchos escritores, al describir a Stallman, tienden a buscar el ángulo religioso. En un artículo de 1998 en Salon.com, titulado "El Santo del Software Libre", Andrew Leonard describe los ojos de Stallman "irradiando el poder de un profeta del Antiguo Testamento". [1] Un artículo de 1999 en la revista [Wired] describe la barba de Stallman como "similar a la de Rasputín", [2] mientras que un pérfil elaborado por el [*London Guardian*] describe la sonrisa de Stallman como la de "un discípulo buscando a Jesús". [3]

Todas estas analogías sirven a un propósito pero terminan siendo insuficientes. Fallan porqué no toman en cuenta el lado vulnerable en la personalidad de Stallman. Observe la mirada de Stallman por un periodo prolongado de tiempo y comenzará a notar un cambio sutil. Lo que parece en principio un intento de intimidar o hipnotizar se revela a si mismo al cabo de una segunda y una tercera observación como un intento fallido para entablar y mantener contacto. Si, como Stallman mismo ha sospechado a veces, su personalidad es el fruto del autismo, o Síndrome de Asperger, sus ojos con seguridad confirman el diagnóstico. Incluso cuando se encuentran en el nivel más alto de intensidad, tienen una tendencia a volverse nublados y distantes, como los ojos de un animal herido preparándose para rendirse ante el fantasma.

Mi primer encuentro con la legendaria mirada de Stallman se remonta a la Convención y Exposición Linuxworld en San José, California, en 1999. Llamada una "fiesta de lanzamiento" para la comunidad de software de Linux, la convención también es conocida por haber sido el evento que reintrodujó a Stallman a los medios de comunicación. Decidido a obtener su parte del crédito, Stallman utilizó el evento para inducir tanto a los espectadores como a los reporteros, en la historia del Proyecto GNU y de sus objetivos políticos.

Cómo un periodista enviado a cubrir el evento, recibí mi propio tutorial de Stallman durante una conferencia de prensa anunciando el lanzamiento de GNOME 1.0, una interfaz gráfica de usuario hecha con Software Libre. Inadvertidamente, toqué una gran cantidad de fibras sensibles cuando le hice la primera pregunta a Stallman mismo: ¿Piensa usted que la madurez de GNOME afectará la popularidad comercial del Sistema Operativo Linux?

"Le pido que por favor deje de llamar al sistema operativo Linux", responde Stallman, sus ojos inmediatamente fijándose en los mios. "El kernel de Linux es solamente una pequeña parte del sistema operativo. Muchos de los programas de software que constituyen el sistema operativo que usted llama Linux no fueron desarrollados por Linus Torvalds en absoluto. Fueron creados por voluntarios del Proyecto GNU, que ofrecieron su propio tiempo para que los usuarios tuvieran un sistema operativo

libre como el que tenemos hoy. No reconocer la contribución de esos programadores es descortés y una tergiversación de la historia. Por eso es que les pido que cuando se refieran al sistema operativo, por favor lo llamen con el nombre correcto, GNU/Linux."

Anotando las palabras en mi cuaderno de reportero, noto un misterioso silencio en la congestionada sala. Cuando finalmente miro hacía adelante, encuentro los ojos de Stallman esperándome, sin parpadear. Tímidamente, un segundo periodista hace una pregunta, asegurándose de usar del término "GNU/Linux" en lugar de Linux. Miguel de Icaza, líder del proyecto GNOME, responde a la pregunta. No es sino hasta la mitad de la respuesta de Icaza que los ojos de Stallman finalmente dejan de fijarse en los mios. En el momento en que lo hacen, un leve temblor recorre mi espalda. Cuando Stallman comienza a reprender a otro reportero por un error de dicción siento una culpable pizca de alivio. Por lo menos no esta mirándome, me digo a mi mismo.

Para Stallman, aquellos momentos cara a cara cumplirían su cometido. Para el momento en que la primera conferencia de LinuxWorld termina, la mayoría de periodistas han aprendido a no usar el término Linux en su presencia, y wired.com esta realizando una historia en la que compara a Stallman con un revolucionario pre Stalinista borrado de los libros de historia por hackers y empresarios reacios a aceptar los objetivos excesivamente políticos del proyecto GNU. [4] Otros artículos siguen a este, y aunque muy pocos periodistas llaman al sistema operativo GNU/Linux en el papel, la mayoría se apresuran a dar crédito a Stallman por lanzar la idea de construir un sistema operativo libre quince años antes.

No me encontraré de nuevo con Stallman sino hasta 17 meses después de estos eventos. Durante ese tiempo, Stallman volverá de nuevo a *Silicon Valley* una vez más para el LinuxWorld de 1999. A pesar de no estar invitado como orador, Stallman aún se las arregla para ser el autor del "comentario del evento". En el momento de aceptar el Premio Linus Torvalds al Servicio Comunitario --un premio bautizado en honor al creador de Linux -- en representación de la *Free Software Foundation*, Stallman dice, "Darle el Premio Linus Torvalds a la Free Software Foundation es un poco como darle el premio Han Solo a la Alianza Rebelde."

Esta vez, sin embargo, los comentarios no hacen mucho escándalo en los medios. En la mitad de la semana Red Hat, Inc. un prominente vendedor de GNU/Linux, sale a la bolsa. Las noticias simplemente confirman lo que muchos reporteros, como yo, ya sospechábamos: "Linux" se ha convertido en una palabra de moda en Wall Street, como "e-commerce" y "punto com". Con el mercado accionario aproximándose al cambio de milenio como una hipérbola acercándose a su asíntota vertical, todo lo que se habla de software libre o código fuente abierto como un fenómeno político es dejado a un lado.

Tal vez es por eso que cuando LinuxWorld continúa sus dos primeras versiones con un tercer LinuxWorld en agosto del 2000, Stallman esta notablemente ausente.

Mi segundo encuentro con Stallman y su mirada de marca registrada ocurre poco después del tercer LinuxWorld. Enterado de que Stallman iba a estar en Silicon Valley, arreglé una entrevista a la hora del almuerzo en Palo Alto, California. El lugar de encuentro parece irónico, no sólo por la reciente no asistencia de Stallman sino por el ambiente en general. Además de Redmond, Washington, muy pocas ciudades ofrecen un homenaje más directo al valor económico del software propietario. Curioso por saber como Stallman, un hombre que ha empleado sus mejores años batallando contra nuestra predilección cultural hacía la avaricia y el egoismo, se comporta en una ciudad donde incluso las habitaciones del tamaño de un garaje tienen precios del orden de medio millón de dólares, conduzco hasta allí desde Oakland.

Sigo las direcciones que Stallman me ha dado, hasta que llegó al centro de operaciones de Art.net, una "colectividad de artistas virtuales" sin ánimo de lucro. Localizado en una casa cubierta por arbustos en la esquina norte de la ciudad, las oficinas de Art.net se encuentran refrescantemente descuidadasDe repente, la idea de Stallman acechando en el corazón de *Silicon Valley* no parece tan extraña después de todo.

Encuentro a Stallman sentado en un cuarto oscuro, tecleando en su computador portátil gris. El voltea hacía mi en el momento en que entro al cuarto, entregándome de un golpe todo el poder de su mirada de 200 vatios. Cuando me ofrece un reconfortante "Hola", yo ofrezco un saludo de vuelta. Antes de que las palabras salgan de mi boca, sus ojos ya han vuelto a la pantalla de su portátil.

"Estoy terminando un artículo acerca del espíritu del hacking", dice Stallman, mientras sus dedos continúan tecleando. "Mire."

Yo observo. El cuarto esta pobremente iluminado, y el texto aparece como letras blancas-verdosas sobre un fondo negro, un esquema de colores opuesto a aquel utilizado por la mayoría de procesadores de palabras, por lo que mis ojos necesitan un momento para ajustarse. Cuando lo hacen, me encuentro a mi mismo leyendo el relato de Stallman de una comida reciente en un restaurante Coreano. Antes de la comida Stallman hace un descubrimiento interesante: la persona encargada de arreglar la mesa ha dejado seis palilos, en lugar de los acostumbrados dos en el puesto de Stallman. La mayoría de comensales de un restaurante habrían ignorado los pares sobrantes, Stallman lo toma como un reto: encontrar una manera de usar los seis palillos al tiempo. Como muchos hacks de software la solución es a la vez inteligente y tonta. De ahí la decisión de Stallman de usar la anécdota como una ilustración.

A medida que leo la historia, siento a Stallman observándome atento. Yo miro de reojo y observo una media sonrisa en su cara, un gesto de orgullo infantil. Cuando alabo el ensayo, mi comentario merece simplemente un leve arqueo de sus cejas.

"Estaré listo para salir en un momento", dice.

Stallman vuelve a escribir en su portátil. Este es gris y cuadrado, no como los elegantes portátiles de última moda, preferidos entre los programadores que asistieron a LinuxWorld. Encima del teclado se encuentra un teclado más pequeño y liviano, un homenaje a las envejecidas manos de Stallman. A finales de los años ochenta, cuando Stallman estaba trabajando en semanas de 70 y 80 horas, escribiendo las primeras herramientas y programas de software libre para el proyecto GNU, el dolor en sus manos se hizo tan insoportable que se vió obligado a contratar un digitador. Hoy en día, Stallman usa un teclado cuyas teclas requieren menos presión que las de un teclado típico.

Stallman tiende a bloquear todos los estímulos externos mientras se encuentra trabajando. Al ver sus ojos fijarse en la pantalla y sus dedos bailar, rápidamente se ve la imagen de dos amigos involucrados en una profunda conversación.

La sesión termina con un par de teclazos fuertes y el lento proceso de desarmar el portátil.

"¿Listo para almorzar?" Pregunta Stallman.

Caminamos hacía mi automóvil. Aduciendo una lesión de tobillo, Stallman cojea lentamente. Stallman culpa a la lesión de un tendón de su pie izquierdo. La lesión tiene tres años y se ha vuelto tan grave que Stallman, un fanático entusiasta de las dazas folclóricas, se ha visto obligado a dejar todas las actividades de baile. "Amo la danza folclórica como parte de mí", lamenta Stallman. "No ser capaz de bailar ha sido una tragedia para mí."

El cuerpo de Stallman permanece como testigo de esta tragedia. La falta de ejercicio ha dejado a Stallman con las mejillas infladas y una barriga que era mucho menos visible hace un año. Uno se da cuenta de que la subida de peso ha sido dramática ya que, cuando Stallman camina, arquea su columna como una mujer embarazada tratando de acomodar una carga otrora desconocida.

La caminata es demorada aún mas por el deseo de Stallman de pararse a oler las rosas, literalmente. Al encontrar un ramillete particularmente hermoso, cosquillea los pétalos mas ocultos con su prodigiosa nariz, aspira profundamente y regresa con un suspiro satisfecho,

"Mmm, *rinofitofilia*", [5] dice, frotando su espalda.

El viaje hasta el restaurante toma menos de tres minutos. Por recomendación de Tim Ney, antiguo director ejecutivo de la Fundación para el Software Libre, he dejado a Stallman escoger el restaurante. A pesar de que muchos periodistas se concentran en el estilo de vida casi monacal de Stallman, lo cierto es que Stallman es un epicuro comprometido en cuanto se refiere a la comida. Uno de los beneficios adicionales de ser un misionero viajero por la causa del Software Libre es la posibilidad de probar comida exquisita de todas partes del mundo. "Visite cualquier ciudad del mundo, y es probable que Richard sepa cuál es el mejor restaurante", dice Ney. "Richard también esta muy orgulloso de saber que esta en el menú, y de ordenar para toda la mesa."

Para la comida de hoy, Stallman ha escogido un restaurante cantonés a dos calles de *University Avenue*, la calle más importante de Palo Alto. La elección esta parcialmente inspirada en la reciente visita de Stallman a China, con una parada para dictar una conferencia en la provincia de Guangdong, sumada a la aversión personal de Stallman hacía la comida Hunanesa y Szechuana, más condimentada. "No soy realmente un fanático de la comida condimentada", admite Stallman.

Llegamos unos minutos después de las 11 a.m. y ya somos sometidos a una espera de veinte minutos. Dada la aversión de los hackers por el tiempo perdido, yo contengo la respiración momentáneamente, esperando una explosión. Stallman, contrariando mis expectativas, toma las noticias con calma.

"Es una lástima no haber podido encontrar nadie más para que nos acompañara", me dice. "Siempre es mas divertido comer con un grupo de gente."

Mientras esperamos, Stallman practica unos pasos de danza. Sus movimientos son tímidos pero hábiles. Discutimos acerca de las últimas noticias. Stallman dice que su lo único que lamenta de no haber asistido a LinuxWorld es no haber podido estar presente en una conferencia que anunciaba el lanzamiento de la Fundación GNOME. Respaldada por Sun Microsystems e IBM la fundación es, de muchas maneras una reivindicación para Stallman, quién ha dicho por mucho tiempo que el software libre y la economía de libre mercado no tienen porqué ser mutuamente excluyentes. Sin embargo, Stallman permanece insatisfecho por el mensaje que finalmente llegó al público.

"De la manera en que fue presentado, las compañias hablaron de Linux sin mencionar para nada al Proyecto GNU", dice Stallman.

Estas decepciones contrastan la cálida respuesta que viene del otro lado del oceano, especialmente de Asia, nota Stallman. Una mirada rápida al itinerario de viajes de Stallman durante el año 2000 habla de la creciente popularidad del mensaje del software libre. Entre sus recientes visitas a India, China y Brasil, Stallman solo ha estado doce de los últimos 115 días en suelo de los Estados Unidos. Sus viajes le han dado la oportunidad de ver como el concepto de software libre se traduce a diferentes idiomas y culturas.

"En la India mucha gente esta interesada en el software libre, porqué lo ven como una manera de crear una infraestructura computacional sin gastar demasiado dinero", dice Stallman. "En China el concepto se ha demorado mucho en ser entendido. Hacer la comparación entre software libre y libertad de expresión es más difícil cuando no hay libertad de expresión. Aún así, el nivel de interés en el software libre durante mi última visita era muy profundo."

La conversación comienza a girar en torno a Napster, la compañía de software de San Mateo, California, que se ha convertido en uno de los temas favoritos de los medios en los últimos meses. La compañía produce una controvertida herramienta de software que les permite a los melómanos buscar y copiar los archivos de música de otros fanáticos de la música. Gracias al poder magnificador de Internet, este programa de "persona a persona" (*peer to peer*) se ha convertido en una caja de música en línea, dándoles a todos los aficionados a la música la posibilidad de escuchar archivos de MP3 en el computador sin pagar derechos, para disgusto de las compañías disqueras.

A pesar de estar basado en software propietario, el sistema de Napster se inspira en la convicción de Stallman, largamente sostenida, de que una vez un trabajo incursiona en el ámbito digital --en otras palabras, una vez hacer una copia deja de ser asunto de duplicar sonidos o duplicar átomos, para convertirse en asunto de duplicar información-- entonces es más difícil restringir el impulso humano de compartir un trabajo. En lugar de imponer restricciones, los ejecutivos de Napster han decido aprovecharse de este impulso. Dándole a los melómanos un lugar central para intercambiar archivos de música, la compañía ha jugado con su habilidad de enviar el tráfico de usuarios resultante hacía otras oportunidades comerciales.

El repentino éxito del modelo de Napster ha asustado a las compañías de discos tradicionales, con muy buenas razones. Sólo unos días antes de mi encuentro con Stallman en Palo Alto, la juez de distrito Marilyn Patel falló a favor de la Asociación de la Industria Disquera de América (*Recording Industry Association of America*) en su moción contra el sistema de compartir archivos. El falló fue posteriormente suspendido por la Novena Corte de Apelaciones de Distrito, pero para principios de 2001 la misma Corte de Apelaciones encontraría que la compañía de San Mateo estaba violando las leyes de derechos de autor, [6] una decisión que la portavoz de la RIAA, Hillary Rosen llamaría mas tarde "una clara victoria de la comunidad de contenido creativo y del legítimo mercado en línea" [7]

Para hackers como Stallman, el modelo de negocios de Napster es preocupante de varias maneras. La velocidad de la compañía para apropiarse de principios hackers gastados por el tiempo, como el de compartir archivos y el de la posesión comunal de la información, mientras que al mismo tiempo venden un servicio basado en software propietario, envían un mensaje distorsionado de manera preocupante. Cómo una persona que de por si tiene suficientes problemas para hacer llegar su propio mensaje, cuidadosamente artículado, a los medios de comunicación, es entendible la reticencia de Stallman en el momento de hablar acerca de la compañía. Sin embargo, Stallman admite haber aprendido una cosa o dos de la faceta social del fenómeno Napster.

"Antes de Napster, yo pensaba que podría ser correcto que la gente redistribuyera privadamente obras de entretenimiento", dice Stallman. "La cantidad de personas a quienes Napster les pareció útil, sin embargo, me dice que el derecho de redistribuir copias, no solo entre vecinos, sino para el público en general, es esencial y por lo tanto no puede ser usurpado."

No ha terminado Stallman de decir esto y la puerta del restaurante se abre y somos invitados adentro por el acomodador. Al cabo de unos pocos segundos, estamos sentados en una esquina del restaurante, adyacente a una gran pared cubierta con un espejo.

La carta del restaurante se dobla como un formulario de compra, y Stallman esta marcando opciones antes incluso de que nos hayan traido agua. "Rollo de camarón frito envuento en nata de fríjol", lee Stallman. "Nata de fríjol. Es una textura tan interesante. Creo que deberíamos ordenarlo."

Este comentario nos lleva a una improvisada discusión sobre la comina China y la reciente visita de Stallman a ese país. "La comida en China es absolutamente exquisita", dice Stallman, a medida que su voz adquiere un toque de emoción por primera vez en toda la mañana. "Tantas cosas diferentes que nunca he visto en EEUU, cosas locales hechas con hongos locales y vegetales locales. Llegó hasta el punto en el que comencé a llevar un diario, solo para llevar la cuenta de cada una de las maravillosas comidas."

La conversación continúa con una discusión sobre la cocina Coreana. Durante el mismo recorrido asiático de Junio del 2000, Stallman hizo una visita a Corea del Sur. Su llegada motivó una pequeña tormenta entre los medios locales, gracias a una conferencia de software Coreano a la que asistió en fundador y presidente de Microsoft, Bill Gates y que se llevó a cabo esa misma semana. Después de ver su foto encima de la de Gates en la primera página del principal periódico de Seúl, Stallman dice que lo mejor del viaje fue la comida. "Probé una taza de naeng myun, que son fideos fríos", dice Stallman. "Eran unos fideos que causaban una sensación muy interesante. La mayoría de lugares no usan exactamente el mismo tipo de fideos para el naeng myun, por lo que puedo decir, con completa certeza, que este fue el naeng myun mas exquisito que jamás haya probado."

El término "exquisito" es un gran elogio, viniendo de Stallman. Se esto, porqué unos momentos después de escuchar a Stallman poetizar acerca de naeng myun, siento sus ojos de rayos láser perforando la parte de arriba de mi hombro derecho.

"Detrás de usted esta sentada la mujer más exquisita", dice Stallman.

Me doy la vuelta para mirar, y sólo alcanzo a echar una ojeada a la espalda de una mujer. La mujer es joven, aproximadamente de 25 años y lleva un vestido blanco. Ella y su compañero están terminando de pagar la cuenta. Cuando ambos se paran de la mesa, para abandonar el restaurante, me entero sin mirar, porqué los ojos de Stallman de repente pierden algo de su intensidad.

"Oh, no," dice. "Se fueron. Y pensar que probablemente no la vuelva a ver nunca."

Tras un breve suspiro, Stallman se recupera. El momento me da una oportunidad para discutir la reputación de Stallman hacía el sexo opuesto. Esta reputación es a veces un poco contradictoria. Varios hackers hablan de la predilección de Stallman por saludar a las mujeres con un beso en el dorso de la mano. [8] Un artículo de Salon.com, aparecido el 26 de mayo del 2000, retrata a Stallman un poco como un hacker libertino. Documentando la relación entre el software libre y el amor libre, la reportera Analee Newitz muestra a Stallman como alguién que rechaza los valores tradicionales de la familia, cuando le dice "Yo creo en el amor, no en la monogamia" [9]

Stallman deja caer ligeramente su menú cuando traigo a colación el tema. "Bueno, la mayoría de los hombres parecen querer sexo y parecen tener una actitud bastante despreciativa hacía las mujeres", dice."Todas las mujeres con las que se relacionan. No lo entiendo en absoluto."

Menciono el pasaje del libro [Open Sources] en el que Stallman confiesa haber querido nombrar el kernel de GNU, de destino incierto, con el nombre de su novia del momento. El nombre de la novia era Alix, un nombre que encajaba perfectamente con la conveción de los desarrolladores de poner una "x" al final de cada nuevo nombre de kernel-- por ejemplo, "Linux". Puesto que ella era administradora de un sistema Unix, Stallman dice que habría sido un tributo aún más sentido. Desafortunadamente, anota Stallman, el desarrollador principal del proyecto del kernel eventualmente

lo llamó HURD. [10] A pesar de que Stallman y la mujer después rompieron, la historia suscita una pregunta casi automática: A pesar de todos las creaciones de los medios, que muestran a Stallman como un fanático, ¿Es Richard Stallman realmente sólo un romántico sin esperanzas, un Quijote errante, atacando molinos corporativos en un esfuerzo por impresionar a una Dulcinea aún no identificada?

"Realmente no estaba tratando de ser romántico", dice Stallman recordando la historia de Alix. "Era más un asunto de provocación. Es decir, era romántico pero a la vez era provocativo. ¿Sabe? Habría sido una sorpresa estupenda."

Por primera vez en toda la mañana, Stallman rie. Yo traigo al tema el asunto de los besos en la mano. "Si, yo hago eso", dice Stallman. "He encontrado que es una manera de ofrecer algo de afecto que muchas mujeres disfrutarán. Es una oportunidad de dar algo de afecto y de ser estimado por ello."

El afecto es un tema constante que aparece repetidamente en la vida de Richard Stallman, y el es dolorosamente abierto al respecto cuando se hacen las preguntas. "Realmente no ha habido demasiado afecto en mi vida, excepto en mi mente", dice. Sin embargo, la discusión rápidamente se vuelve incómoda. Después de algunas respuestas con monosílabos, Stallman finalmente levanta su menú, dando por terminada la pesquisa.

"¿Le gustaría algo de shimai?" pregunta.

Cuando llega la comida, la conversación se mueve entre los platos que van llegando. Discutimos la frecuentemente citada propensión de los hackers por la comida China, las salidas semanales a comer al Barrio Chino de Boston durante los días de Stallman como programador de planta del laboratorio de IA, y la lógica que subyace el idioma Chino y su sistema de escritura. Cada comentario de mi parte provoca una respuesta bien informada de parte de Stallman.

"Oí a algunas personas hablando Shangaiés la última vez que estuve en China", dice Stallman. "Era interesante de oir. Sonaba bastante distinto [del mandarín]. Hice que me dijeran palabras con raíces similares en Mandarín y en Shangaiés. En algunos casos es posible ver las similitudes, pero yo me preguntaba si los tonos eran similares. No lo son. Eso me parece interesante porqué hay una teoría que dice que los tonos evolucionaron de sílabas adicionales que se perdieron y fueron reemplazados. Si eso es cierto, y he visto teorías que dicen que eso sucedió en tiempos históricos, entonces los dialectos deben haber divergido antes de la pérdida de estas sílabas finales."

El primer plato, nabos fritos a la sartén, ha llegado. Tanto Stallman en como yo nos tomamos un momento para partir las grandes tortas rectangulares, que huelen como repollo hervido pero saben como rebanadas de papas fritas en tocino.

Decido retormar el tema de la falta de adaptación, preguntándome si los años de adolescencia de Stallman lo condicionaron para abogar por causas impopulares, mas notablemente su batalla contra corriente, desde 1994 para lograr que los usuarios de computadores y los medios reemplazaran el término popular "Linux" por "GNU/Linux".

"Creo que si me ayudó", dice Stallman, mientras mastica un bocado. "Nunca he entendido lo que la presión de los compañeros hace a otras personas. Creo que la razón era que yo estaba tan irremediablemente rechazado que, para mi, no había nada que ganar si intentaba seguir cualquiera de las modas. Todo habría sido exactamente igual. Seguiría igual de rechazado, por lo que no lo intente."

Stallman anota su gusto musical como un ejemplo clave de sus tendencias contrarias a la masa. Cuando era una adolescente, y la mayoría de sus compañeros de clase escuchaban Motown y Acid Rock, Stallman prefería la música clásica. El recuerdo lleva a un extraño y jocoso episodio de los años de bachillerato de Stallman. Luego de la aparición de los Beatles en el Show de Ed Sullivan, en 1964, la mayoría de los compañeros de clase de Stallman se precipitaron a comprar los últimos álbumes y sencillos de los Beatles. Allí y en ese momento, dice Stallman, tomó la decisión de sabotear a los Cuatro de Liverpool.

"Me gustaba alguna de la música popular anterior a los Beatles", dice Stallman. "Pero no me gustaban los Beatles. Me disgustaba especialmente la manera salvaje en que la gente reaccionaba ante ellos. Era algo así como: ¿Quién iba a hacer un montaje de los Beatles que adulara mas a los Beatles?"

Cuando su sabotaje a los Beatles no funcionó, Stallman buscó otras formas de hacer evidente la mentalidad de rebaño de sus compañeros. Stallman dice que considero brevemente la posibilidad de formar él mismo una banda de rock dedicada a satirizar a los Beatles.

"La quería llamar *Tokyo Rose and the Japanese Beetles*." [11]

Dada su pasión actual por la música folclórica internacional, le pregunté a Stallman si sentía una afinidad similar por Bob Dylan y los otros músicos folclóricos de comienzos de los sesentas. Stallman sacude su cabeza. "Me gustaban Peter, Paul y Mary", dice. "Eso me recuerda un gran *filk*."

Cuando le pido una definición de "*filk*, Stallman me explica el concepto. Un filk, me dice, es una canción popular cuya letra se alteró para ser reemplazada por unas letras de parodia. El proceso de escribir "filks" es una actividad popular entre los hackers y los aficionados a la ciencia ficción. La lista de filks clásicos incluye "*On Top of Spaghetti*", una reescritura de "*On Top of Old Smokey*", y "Yoda", una modificación tipo Guerra de las Galaxias de la canción de los Kinks, "Lola", escrita por el maestro de los filks "Weird" Al Yankovic.

Stallman me pregunta si estaría interesado en escuchar el filk folclórico. En el momento en que digo que si, la voz de Stallman comienza a cantar en un tono inesperadamente claro:

> How much wood could a woodchuck chuck, If a woodchuck could chuck wood? How many poles could a polak lock, If a polak could lock poles? How many knees could a negro grow, If a negro could grow knees? The answer, my dear, is stick it in your ear. The answer is to stick it in your ear. [12]

La canción termina, y los labios de Stallman se doblan en otra media sonrisa infantil. Yo observo de reojo en la mesas cercanas. Las familias asiáticas que disfrutan de su almuerzo dominical le prestan poca atención al cantante barbado entre ellos. [13] Después de unos momentos de vacilación, finalmente yo también sonrio.

"¿Quieres ese último bocado?" Pregunta Stallman, con los ojos iluminados. Antes de que pueda dañarle la línea, Stallman toma el bollo de maíz con sus dos palillos y lo levanta orgulloso. "Tal vez deba ser yo quien tome el último bocado" dice.

Tras terminar la comida, nuestra conversacion adquiere el ritmo de una entrevista normal. Stallman reclina su asiento y toma una taza de té entre sus manos. Continuamos hablando acerca de Napster y su relación con el movimiento del software libre. ¿Deberían ser extendidos los principios del software libre a áreas similares como la publicación de música? Le preguntó.

"Es un error copiar las respuesta de un sitio a otro", dice Stallman, comparando la canciones con programas de software. "La manera correcta es mirar cada tipo de trabajo y ver a que conclusión se llega."

Cuando se refiere a los trabajos protegidos por derechos de autor, Stallman dice que él divide al mundo en tres categorías. La primera categoría involucra las obras "funcionales" -- programas de software, diccionarios y libros de texto. La segunda categoría tiene que ver con obras que serían mejor descritas como "testimoniales" --trabajos científicos y documentos históricos. Estas obras cumplen una labor que se vería perjudicada si lectores o autores tuvieran la posibilidad de modificarlos libremente. La última categoría se refiere a las obras de expresión personal --diarios, periódicos, y autobiografías. Modificar esos documentos sería alterar las impresiones o el punto de vista de una persona- una acción que Stallman considera éticamente injustificable.

De las tres categorías, la primera debería dar a los usuarios el derecho ilimitado a crear versiones modificadas, mientras que la segunda y la tercera deberían regular ese derecho de acuerdo con el autor original. Sin importar la categoría, sin embargo, la libertad de copiar y redistribuir para fines no comerciales debe permanecer inalterada en todo momento, insiste Stallman. Si eso implica darle a los usuarios de Internet el derecho a generar cien copias de un artículo, canción, o libro, para después enviarlas por correo electrónico a cien extraños, pues sea. "Es claro que la redistribución privada esporádica debe ser permitida, porque solo un estado policía puede detener eso", dice Stallman. "Es antisocial interponerse entre la gente y sus amigos. Napster me ha convencido de que también debemos permitir, tenemos que permitir, incluso la distribución no comercial al público, solo por diversión. Porqué hay tanta gente que lo quiere hacer y a quienes les parece tan útil,"

Cuando le pregunto si los estrados judiciales aceptarían una visión tan permisiva, Stallman me interrumpe.

"Esa es la pregunta incorrecta", dice. "Quiero decir, ahora has cambiado el tema completamente de uno de ética a otro de interpretación de las leyes. Y esas son dos preguntas totalmente diferentes del mismo campo. No tiene ninguna utilidad saltar de una a la otra. Los estamentos judiciales interpretarán las leyes de manera dura, porqué esa es la manera en que esas leyes han sido compradas por quienes publican."

El comentario proporciona una luz acerca de la filosofía política de Stallman: solo porqué el sistema legal actualmente respalda la capacidad de las empresas de tratar los derechos de autor como el equivalente en software de la tierra, eso no significa que los usuarios de computadores deban jugar de acuerdo a esas mismas reglas. La libertad es un asunto ético, no un asunto legal. "Estoy viendo mas allá de lo que son las leyes actuales, hacia lo que deberían ser", dice Stallman. "No estoy tratando de rehacer la legislación. Estoy pensando en lo que la ley debería hacer. Considero que la prohibición legal de compartir copias con tus amigos es el equivalente moral de Jim Crow [14]. No merece respeto."

La mención de Jim Crow provoca otra pregunta. ¿Que tanta influencia o inspiración recibe Stallman de líderes políticos del pasado? Como el movimiento de los derechos civiles de las décadas del 50 y el 60, su intento de generar cambios a nivel social se basa en un llamado a valores atemporales: libertad, justicia y juego limpio.

Stallman divide su atención entre mi analogía y un mechón de cabello particularmente enredado. Cuando extiendo la analogía hasta el punto en el que comparo a Stallman con el Dr. Martin Luther King, Jr., Stallman, me interrumpe.

"No tengo su nivel, pero si juego el mismo juego", dice.

Le sugiero a Malcolm X como otro punto de comparación. Como el otrora vocero de la Nación Islámica, Stallman ha creado una reputación de atraer la controversia, enemistarse con aliados potenciales, y predicar un mensaje que favorece la autosuficiencia sobre la integración cultural.

Stallman rechaza la comparación. "Mi mensaje es mas cercano al del doctor King. Es un mensaje de firme condena a ciertas prácticas que maltratan al otro. No es un mensaje de odio hacía nadie. Y no esta orientado a un grupo reducido de gente. Yo invito a cualquiera a valorar la libertad, y a tener libertad.

Aún así, una actitud sospechosa hacía las alianzas políticas permanece como un rasgo fundamental de la personalidad de Stallman. En el caso de su bien publicado disgusto por el término "software de fuente abierta" (*Open Source*), su falta de voluntad para participar en proyectos recientes de coalición parece entendible. Como un hombre que ha dedicado las dos últimas décadas a luchar por la causa del software libre, el capital político de Stallman esta profundamente invertido en el término. Así, comentarios como aquel de "Han Solo" en LinuxWorld de 1999 no han hecho más que reforzar la reputación de Stallman en la industria del software. Una reputación que lo describe como un hombre descontento, que no esta dispuesto a moverse al ritmo de las modas políticas o de mercadeo.

"Yo admiro y respeto a Richard por todo el trabajo que ha hecho", dice el presidente de Red Hat, Robert Young, resumiendo la naturaleza política paradójica de Stallman. "Mi única crítica es que a veces Richard trata a sus amigos peor que a sus enemigos."

La renuencia de Stallman a buscar alianzas aparece igual de asombrosa cuando se consideran los intereses políticos de Stallman fuera del movimiento de software libre. Visite las oficinas de Stallman en MIT, y encontrará instantáneamente un depósito de artículos noticiosos de izquierda que se refieren a abusos de los derechos civiles alrededor del mundo. Visite su página web, y encontrará diatribas contra el Digital Millennium Copyright Act, la guerra contra las drogas y la Organización Mundial de Comercio.

Dadas sus tendencias de activista, yo pregunto, ¿Por qué no a buscado Stallman ser oido de manera mas amplia? ¿Por qué no ha usado su prominencia en el mundo hacker como una plataforma para aumentar, en lugar de reducir, su voz política?

Stallman deja caer su enredado cabello y contempla la pregunta por un momento.

"Dudo en si exagerar la importancia de este pequeño charco de libertad", dice. "Porqué las áreas mas conocidas y convencionales del trabajo por la libertad y por una sociedad mejor son tremendamente importantes. No podría decir que el software libre es tan importante como ellas. Es la responsabilidad que yo asumí, porqué cayó en mi regazo y vi una manera en la que yo podía hacer algo al respecto. Pero, por ejemplo, terminar con la brutalidad policial, acabar la guerra contra las drogas, exterminar las formas de racismo que aún conservamos, ayudar a todos a tener una vida confortable, proteger los derechos de quienes realizan abortos, protegernos de la teocracía, estos son temas tremendamente importantes, mucho más importantes que lo que hago yo. Solo quisiera saber como hacer algo al respecto."

Una vez más, Stallman presenta su actividad política como una función de su convencimiento personal. Dada la cantidad de tiempo que le ha tomado desarrollar y ver crecer las doctrinas fundamentales del movimiento para el software libre, Stallman duda en el momento de involucrarse con otras causas o movimientos que lo puedan llevar a territorio desconocido.

"Me gustaría saber como hacer una diferencia importante en esos asuntos importantes, porqué me sentiría tremendamente orgulloso si pudiera hacerlo, pero es muy difícil y mucha gente que es probablemente mejor que yo ha estado intentándolo, llegando solo hasta cierto punto", dice. "Pero como yo lo veo, mientras otras personas se estaban defendiendo contra estas amenazas enormes y evidentes, yo vi otra amenaza que no estaba siendo vigilada. Y fuí a formar la defensa contra esa amenaza. Puede no ser una amenaza tan grande, pero yo era el único allí."

Finalmente, Stallman sugiere pagar la cuenta. Antes de que el mesero pueda llevársela, sin embargo, Stallman saca un billete de un dólar coloreado de blanco y lo pone en la pila. El billete es tan evidentemente falso que no puedo evitar tomarlo y leerlo. Claro, es una falsificación. En lugar de llevar la imagen de George Washington o Abraham Lincon, el frente del billete presenta la imagen de un cerdo de caricatura. En lugar de decir *United States of America*, el letrero sobre el cerdo dice *"United Swines of Avarice"* (Seres detestables unidos de la avaricia). El billete es de cero dólares, y cuando el mesero recoje el dinero, Stallman se asegura de halarle la manga.

"Añadí un cero de más a su propina", dice Stallman, y una vez más media sonrisa aparece en sus labios.

El mesero, sin comprender, o engañado por el billete, sonríe y se aleja rápidamente.

"Creo que eso significa que somos libres para irnos", dice Stallman.

Notas

[1] Ver Andrew Leonard, *"The Saint of Free Software"*, [Salon.com] (Agosto 1998). http://www.salon.com/21st/feature/1998/08/cov_31feature.html

[2] Ver Leander Kahney, *"Linux's Forgotten Man"* (El Hombre Olvidado de Linux), [*Wired News*] (Marzo 5, 1999). http://www.wired.com/news/print/012941829100.html

[3] Ver *"Programmer on moral high ground; Free software is a moral issue for Richard Stallman believes in freedom and free software."*(Programador en el curubito moral; El Software Libre es un asunto de moral pues Richard Stallman cree en la libertad y el software libre [London Guardian] (Noviembre 6, 1999). Esta es solamente una pequeña muestra de las comparaciones religiosas. Hasta hoy la comparación más extrema ha sido la hecha por Linus Torvalds quién, en su autobiografía --ver Linus Torvalds y David Diamond, [*Just For Fun: The Story of an Accidentaly Revolutionary*] (Sólo por Diversión: La Historia de un Revolucionario Accidental) (HarperCollins Publishers, Inc., 2001): 58--escribe "Richard Stallman es el Dios del Software Libre." Una mención de honor va para Larry Lessig, quién, en una descripción de pie de página de su libro --ver Larry Lessig, [*The Future of Ideas*](El Futuro de las Ideas) (Random House, 2001): 270 -- asimila a Stallman con Moisés:

> … como con Moisés, fue otro líder, Linus Torvalds, quién finalmente llevó al movimiento a la tierra prometida, facilitando el desarrollo de la parte final del rompecabezas del Sistema Operativo. Como Moisés, Stallman es respetado y criticado por aliados dentro del movimiento. El es el líder inflexible, y por ello inspirador, de un aspecto crítico de la cultura moderna. Tengo un respeto profundo hacía los principios y el nivel de compromiso de este extraordinario individuo, y también guardo un gran respeto hacía aquellos lo suficientemente valientes como para cuestionar su manera de pensar y después soportar su ira.

En una última entrevista, le pregunté a Stallman que pensaba acerca de las comparaciones religiosas. "Algunas personas me comparan con un profeta del Antiguo Testamento. La razón es que los profetas afirmaban que algunas prácticas eran incorrectas. Ellos no hacían concesiones con asuntos de moral, no podían ser comprados y eran usualmente despreciados."

[4] Ver Leander Kahney, "*Linux's Forgotten Man*" (El Hombre Olvidado de Linux), [Wired News] (Marzo 5, 1999). http://www.wired.com/news/print/012941829100.html

[5] En ese momento, pensé que Stallman se estaba refiriendo al nombre científico de la flor. Meses después supe que *rhinofitofilia* era una referencia cómica a la actividad, i.e., Stallman metiendo su nariz dentro de una flor y disfrutando el momento. Para conocer otro incidente cómico de Stallman con las flores visite: http://www.stallman.org/texas.html

[6] Cecily Barnes y Scott Ard, *"Court Grants Stay of Napster Injunction"* (La Corte mantiene delito de Napster), [News.com] (Julio 28, 2000). http://news.cnet.com/news/0-1005-200-2376465.html

[7] Ver *"A Clear Victory for Recording Industry in Napster Case"*, ("Una Clara Victoria para la Industria Disquera en el caso Napster") comunicado de prensa de la RIAA (Febrero 12 de 2001). http://www.riaa.com/PR_story.cfm?id=372

[8] Ver Mae Ling Mak, "Mae Ling's Story" (La historia de Mae Ling) (Diciembre 17, 1998). http://www.crackmonkey.org/pipermail/crackmonkey/1998q4/003006.htm Hasta ahora, Mak es la única persona que he encontrado dispuesta a hablar respecto a esta práctica, a pesar de que he escuchado esto de otras fuentes femeninas. Mak, a pesar de expresar un rechazo inicial, finalmente logro dejar de lado su desconfianza y bailar con Stallman en LinuxWorld en 1999. http://www.linux.com/interact/potd.phtml?potd_id=44

[9] Ver Annalee Newitz, *"If Code is Free Why Not Me?"* (Si el Código es Libre, ¿Por qué no Yo? [Salon.com] (Mayo 26, 2000). http://www.salon.com/tech/feature/2000/05/26/free_love/print.html

[10] Ver Richard Stallman, "The GNU Operating System and the Free Software Movement" (El Sistema Operativo GNU y el Movimiento del Software Libre), [Open Sources] (O'Reilly & Associates, Inc., 1999): 65.

[11] La Rosa de Tokio y los Escarabajos Japoneses

[12] La traducción de la canción no tiene ningún sentido, puesto que se pierden los juegos de palabras y la rima:

> ¿Que tanta madera podría una marmota morder, si una marmota pudiera morder madera? ¿Cuántos postes podría un polak bloquear, si un polack pudiera bloquear postes? ¿Cuántas rodillas le podrían crecer a un negro, si las rodillas le pudieran crecer a un negro? La respuesta, querida es ponlo en tu oreja. La respuesta es ponerlo en tu oreja.

[13] Para mas filks de Stallman, visite http://www.stallman.org/doggerel.html. Para oír a Stallman cantando "La Canción del Software Libre", visite http://www.gnu.org/music/free-software-song.html

[14] El término Jim Crow se originó hacía 1830, y se convirtió en un símbolo de la segregación racial aplicada en los estados del sur de los Estados Unidos.

El termino "Leyes de Jim Crow" se refería a todo el aparato legislativo que hacía posible la segregación racial y la separación de razas en los lugares públicos. (Nota del traductor)

Capítulo 6. La comuna de Emacs

El laboratorio de Inteligencia Artificial (IA) de 1970 era desde donde se le viera un lugar especial. Proyectos de frontera e investigadores de primera categoría le dieron una posición eleveda en el mundo de la ciencia de los computadores. Así mismo, la cultura hacker interna y sus políticas anárquicas llevaron a una mística revolucionaria. Solo después, cuando muchos de los científicos y superestrellas del software del laboratorio habían partido, se darían cuenta los hackers del mundo único y efímero en el que cual una vez habían vivido.

"Era un poco como el Jardín del Edén", dice Stallman, resumiendo al laboratorio y a su ethos de compartir de software en un artículo de 1998 en Forbes. "No se nos había ocurrido no cooperar". [1]

Tales descripciones mitológicas, a pesar de ser extremas, subrayan un hecho importante. El piso noveno de 545 Tech Square era más que un lugar de trabajo para muchos. Para hackers como Stallman, esto era el hogar.

La palabra "hogar" es un término pesado en el léxico de Stallman. En una crítica a su padres, Stallman, hasta el día presente, se niega a aceptar cualquier otro hogar antes de Currier House, el dormitorio en el cual vivió durante sus días en Harvard. También se conoce que el describe la partida de ese hogar en términos tragicómicos. Una vez, mientras describía su años en Harvard, Stallman dijo que su único resentimiento era que lo habían echado. No fue hasta que le pregunté a Stallman que precipitó su salida, que me di cuenta que había entrado en una de sus frases clásicas.

"En Harvard tenían la política de que si usted pasaba demasiadas clases ellos le pedían que partiera", dice Stallman.

Sin dormitorio ni deseos de regresar a New York, Stallman siguió un camino iluminado por Sussman, y muchos otros hackers antes de el. Cuando ingresó a MIT como un estudiante graduado, Stallman rentó un apartamento en el cercano Cambridge pero muy pronto vio al laboratorio de IA como su hogar *de facto*. En un discurso de 1986, Stallman recordó las memorias del laboratorio de IA durante ese período:

> Yo habré vivido más dentro del laboratorio que la mayoría de la gente, porque por alguna razón cada uno o dos años no tenía apartamento y entonces vivía unos cuantos meses en el laboratorio. Siempre lo encontré bastante cómodo, así como agradable y frío en el verano. Pero no era extraño encontrar gente quedándose dormida dentro del laboratorio, por el entusiasmo; uno se quedaba despierto tanto como pudiera *hack*eando, porque uno no quería detenerse. Y cuando uno estaba completamente exhausto, uno escalaba a la más próxima superficie horizontal suave. Una atmósfera muy informal. [2]

La atmósfera hogareña del laboratorio podía ser un problema algunas veces. Lo que algunos veían como un dormitorio, otros los veían como un guarida del opio electrónico. En el libro de 1976, [Computer Power and Human Reason], el investigador del MIT Joseph Weizenbaum ofreció una crítica mordaz de los "*computer bums*", término acuñado por Weizenbaum para los hackers que poblaban los cuartos de computadores como el laboratorio de IA. "Sus ropas arrugadas, su pelo sin bañar y sus caras sin afeitar, y su pelo enmarañado testificaban que ellos eran indiferentes a sus cuerpos y al mundo en el que se movían", escribe Weizenbaum. "[Los *computer bums*] existían, al

menos cuando estaban conectados, solo a través y para los computadores". [3]

Después de un cuarto de siglo después de su publicación, Stallman todavía se le erizan los pelos cuando oye la descripción de "*computer bum*" de Weizenbaum, discutiendola en tiempo presente como si Weizenbaum mismo estuviera en el cuatro. "El quiere que la gente sea tan solo profesionales, haciéndolo por el dinero y queriéndose ir y olvidarse del asunto lo más pronto posible", dice Stallman. "Lo que el ve como estado normal de las cosas yo lo veo como una tragedia".

La vida de *hacker*, sin embargo, no estaba excenta de la tragedia. Stallman caracteriza su transición de hacker de fin de semana a morador de tiempo completo en el laboratorio de IA como una serie de dolorosos infortunios que solo podian ser remediados a través de la euforia de *hack*ear. Como Stallman lo dice, el primer infortunio fue su graduación de Harvard. Ansioso de continuar sus estudios en física, Stallman entró como estudiante graduado en el MIT. La escogencia de universidad fue natural. No solo le dio a Stallman la oportunidad de seguir los pasos de los grandes alumnos del MIT: William Shockley (’36), Richard P. Feynman (’39), y Murray Gell-Mann (’51), también lo puso dos millas más cerca del laboratorio de IA y de su nuevo computador PDP-10. "Mi atención iba hacia la programación, pero pensé, bueno, tal vez pueda hacer ambas", dice Stallman.

Trabajando en los campos de la ciencia de nivel graduado en el día y programando en los confines monásticos del laboratorio de IA en la noche, Stallman trató de adquirir un balance perfecto. El *folcrum* de su *geek teeter-totterr* fueron sus salidas semanales con la tropa de danza folcrórica, su única actividad social que le garantizaba una interacción módica con el sexo opuesto. Hacia el final del primer año en el MIT, sin embargo, el desastre golpeó. Una lesión en la rodilla forzó a Stallman a renunciar a la tropa. Al principio vió la lesión como un problema temporal, dedicando el tiempo disponible que el hubiera gastado en bailar a trabajar en el laboratorio de IA aún más. Al final del verano, cuando la rodilla todavía dolía y las clases volvían a comenzar, Stallman comenzó a preocuparse. "Mi rodilla no mejoraba", Stallman recuerda, "lo que significaba que debía detener la danza completamente. Yo estaba destrozado".

Sin dormitorio y sin baile, el universo social de Stallman hizo implosión. Como un astronauta experimentando los efectos de la gravedad cero, Stallman descubrió que su habilidad de interactuar con no *hackers*, especialmente mujeres no *hackers* se había atrofiado significativamente. Después de 16 semanas en el laboratorio de IA, la autoconfianza que había acumulado durante los cuatro años en Harvard estaba virtualmente desaparecida.

"Yo sentía basicamente que había perdido mi energía", recuerda Stallman. "Yo había perdido al energía de hacer cualquier cosa excepto lo que me tentaba más inmediatamente. La energía de hacer cualquier otra cosa se había ido. Yo estaba en un total desbalance".

Stallman se retiró del mundo aún más, enfocándose enteramente en su trabajo en el laboratorio de IA. En Octubre de 1975, el se retiraba del MIT, para no volver más. El *hack*eo de software, que alguna vez había sido un hobby, se convirtió en su llamada.

Regresando a tal período, Stallman ve la transición de estudiante de tiempo completo a *hacker* de tiempo completo como inevitable. Más temprano que tarde, el cree, la llamada de sirena del *hack*eo de computadores habría superado su interés en otros asuntos profesionales. "Con las matemáticas y la física, nunca hubiera encontrado una forma de contribuir", dice Stallman, recordando su esfuerzo anterior a la lesión de la rodilla". "Yo hubiera estado orgulloso de hacer avances en cualquiera de esos campos, pero nunca encontré una forma de hacer esto. No sabía donde comenzar. Con el software, vi inmediatamente como escribir cosas que correrían y serían útiles. El placer de tal conocimiento me llevó a querer hacerlo más".

Stallman no fue el primero en igualar a el *hack*eo con el placer. Muchos de los *hackers* que trabajaban en el laboratorio de IA poseían hojas de vida académicas similares. La mayoría de ellos habían venido a alcanzar grados en matemáticas o ingeniería eléctrica solo para rendir su carrera académicas y sus ambiciones profesionales con la alegría total que provenía de resolver problemas nunca antes tratados. Como Santo Tomás de Aquino, el escolástico conocido por trabajar durante largo tiempo en su Summa Teológica, que alcanzaba visiones espirituales, los programadores alcanzaban estados trascendentales internos a través de total concentración mental y cansancio físico. A pesar de que Stallman desprecia las drogas, como la mayoría de *hackers*, el disfrutaba de la *alta* (en inglés *high*) que venía cerca del final de una jornada de codificación de 20 horas.

Tal vez la sensación más placentera, fue el sentimiento de plenitud personal. Cuando se trataba de *hack*ear, Stallman era un natural. Una infancia llena de jornadas de estudio hasta tarde en la noche, le dio la habilidad de trabajar durante largas horas con poco sueño. Como un rechazado social desde los 10 años, el tenía poca dificultad en trabajar solo. Como un matemático con un el don incorporado para la lógica y la previsón Stallman poseía la habilidad de rodear las barreras de diseño que dejaban a muchos *hackers* dando vueltas en círculo.

"El era especial" recuerda Gerald Sussman, un miembro de la facultad del MIT y un ex-investigador en el laboratorio de IA. Describe a Stallman como "un pensador claro y un diseñador claro", Sussman empleó a Stallman como un asistente de investigación de un proyecto en 1975. El proyecto era complejo, involucrando la creación de un programa de IA que pudiera analizar diagramas de circutios. No sólo involucraba un dominio experto de Lisp, un lenguaje de programación construido específicamente para las aplicaciones de IA, sino que también requería la compresión de como un humano se aproximaría a la misma tarea.

Cuando no estaba trabajando en proyectos oficiales como el programa de Sussman de análisis automático de circuitos, Stallman dedicaba su tiempo a proyectos mascota. Era parte del mejor interés de un *hacker*, el mejorar la infraestructura de software del laboratorio, y un uno de los proyectos mascota más grandes de Stallman durante este período fue el programa de edición del laboratorio TECO.

La historia del trabajo de Stallman en TECO durante 1970 está inextrincablemente unida con el liderazgo del movimiento del software libre. Fue un paso significativo en la historia de la evolución de los computadores, tanto que una pequeña recapitulación de tal evolución es necesaria. Durante los decenios de 1950 y 1960, cuando los computadores primero aparecían en las universidades, la programación de computadores era un asunto increíblemente abstracto. Para comunicarse con la máquina, los programadores crearon una serie detarjetas perforadas, con cada tarjeta representando un comando individual de software. Los programadores dejaban las tarjetas al administrador central de la máuqina quien después las insertaba, una a una, dentro de la máquina, esperando que la máquina perforara un nuevo conjunto de tarjetas perforadas, las cuales el programador decifraba como salida. Este proceso, conocido como "procesamiento en bloque" era engorroso y lento. También era susceptible de abusos de autoridad. Uno de los factores motivantes detrás de la aversión visceral a la centralización por parte de los *hackers* era el poder esgrimido por los operadores de sistema quienes dictaminaban que tareas tenían prioridad.

En 1962, los científicos de la computación y los *hackers* involucrados en el proyecto MAC del MIT , un proyecto pionero en el laboratorio de IA, tomaron pasos para aliviar esta frustación. El tiempo compartido, originalmente conocido como "tiempo robado", hizo posible a múltiples programas tomar ventaja de las capacidades operativas de la máquina. Las interfaces de teletipo hicieron posible comunicarse con la máquina, no a través de una serie de tarjetas perforadas sino a través de texto. Un programador escribía los comandos y leía la salida generada por la máquina línea a línea.

Durante los últimos años de 1960, el diseño de interfaz dio saltos adicionales. En una famosa conferencia de 1968, Doug Engelbart, un científico que trabajaba en en el Instituto de Investigación de Stanford, reveló un prototipo de las modernas interfaces gráficas. Añadiendo un televisor al computador y adicionado una aparato apuntador que Engelbarg llamaba un " ratón", el científico creo un sistema aún más interactivo que el sistema de tiempo compartido del MIT. Tratando el dispositivo de video como una impresora de alta velocidad, el sistema de Engelbart le dió al usuario la habilidad de mover el cursor alrededor de la pantalla y ver la posición del cursor actualizada por el computador en tiempo real. El usuario tenía súbitamente la habilidad de posicionar texto en cualquier lugar de la pantalla.

Tales innovaciones tomarían otras dos décadas en hacer su debut en el territorio comercial. Sin embargo, en la década de 1970, las pantallas de video habían empezado a reemplazar los teletipos como terminales de despliegue, creando el potencial de la pantalla completa opuesto a la capacidad de edición línea por línea.

Uno de los primeros programas que tomó ventaja de la capacidad de pantalla completa fue TECO del laboratorio de IA del MIT. La abreviación proviene de Editor de Texto y COrrector (en inglés *Text Editor y COrrector*), el programa había sido actualizado por programadores desde un editor de teletipo para la máquina PDP-6 del laboratorio. [4]

TECO fue una mejora sustancial sobre editores antiguos, pero sin embargo tenía sus limitaciones. Para crear y editar un documento, un programador tenía que escribir una serie de comandos de software especificando cada edición. Era un proceso abstracto. En contraste con los modernos procesadores de palabra, que actualizan el texto con cada tecla, TECO exigía que el usuario ingresará una serie de instrucciones de edición seguido de una secuencia de "fin de comando", solo para cambiar el texto. Con el tiempo, un *hacker* se hacía lo suficientemente eficiente para escribir documentos enteros en modo de edición, pero como Stallman indicaría después, el proceso requería "la habilidad mental del ajedrez a ciegas" [5]

Para facilitar el proceso, los *hackers* del laboratorio de IA habían construido un sistema que desplegaba modos de "fuente" y "resultado" en una pantalla dividida. A pesar de este *hack* innovador, cambiar entre modo y modo era una molestia.

TECO no era el único editor de pantalla completa que flotaba alrededor del mundo de los computadore en ese tiempo. Durante una visita al Laboratorio de Inteligencia Artificial de Stanford en 1976, Stallman encontró un programa de edición llamado E. El programa tenía una característica interna, que permitía al usuario actualizar el texto después de que pulsara cada tecla. En el lenguaje de la programación de 1970, E fue uno de los primeros y rudimentario editores WYSIWYG. Abreviación para "lo que ves es lo que obtienes" (en inglés "*what you see is what you get*") WYSIWYG significaba que un usuario podía manipular el archivo moviéndose a través del texto desplegado, en contraste con trabajar con un programa de edición a través de comandos. [6]

Impresionado por este *hack*, Stallman buscó maneras de expandir la funcionalidad de TECO en una forma similar al regreso al MIT. El encontró una característica de TECO llamada Control-R, escrita por Carl Mikkelson y nombrada a raíz de la combinación de dos teclas que activaba su funcionamiento. El *hack* de Mikkelson cambiaba a TECO de su modo abstracto de ejecución de comandos hacia un modo más intuitivo de tecla a tecla. Stallman revisó la característica de una forma sútil pero significativa. El hizo posible activar algunos comandos de cadena o " macros," usando otras combinaciones de dos teclas. Donde los usuarios habían entrado solo cadenas de comandos y las descartaban después de ingresarlas, el programa de Stallman hizo posible salvar trucos de macro en un archivo y llamarlos después a voluntad. El *hack* de Mikkelson había llevado TECO al nivel de un

editor WYSIWYG. El *hack* de Stallman lo había llevado al nivel de un editor WYSIWYG programable por el usuario. "Eso fue realmente un avance", dice Guy Steele, un compañero *hacker* del AI en ese tiempo. [7]

Debido a la propia cosecha de Stallman, el *hack* de macros trajo una una explosión de innovación futura. "Todas las personas y su hermano estaban escribiendo su propia colección de comandos de edición, un comando para cada cosa que el típicamente quisiera hacer", Stallman recuerda. "La gente se pasaba luego las macros y las mejoraba, haciéndolas más poderosas y general. La colección de redefiniciones gradualmente se volvieron programas en sí mismos". [8]

Tanta gente encontró las innovaciones de las macros útiles y las incorporó de sus programas de TECO que el editor TECO se volvió secundario en comparación con la macromanía que había despertado. "Comenzamos a categorizarlo mentalmente como un lenguaje de programación en vez de un editor", dice Stallman. Los usuarios experimentaban su propio placer modificando el software e intercambiando nuevas ideas. [9]

Dos años después de la explosión, la rata de innovación comenzó a exhibir efectos de borde peligrosos. El crecimiento explosivo había provisto una excitante validación de la aproximación colaborativa de los programadores, pero también había conducido a una complejidad excesiva. "Teníamos un efecto de Torre de Babel" dice Guy Steele.

El efecto amenazaba matar el espíritu que lo había creado, dice Steele. Los *hackers* habían diseñado al ITS para facilitar la habilidad de los programadores de compartir conocimiento y mejorar el trabajo de cada uno. Esto significaba ser capaz de sentarse en otro escritorio de un programador, abrir el trabajo del programa y hacer comentarios y modificaciones directamente sobre el software. "Algunas veces la manera más fácil de mostrar como programar o encontrar errores a alguíen, era simplemente sentarse en la terminal y hacerlo por él", explica Steele.

La característica de las macros, después de su segundo año, comenzó a impedir esta capacidad. En su ansia por abrazar las nueva capacidades de pantalla completa, los *hackers* habían configurado sus versiones de TECO hasta el punto donde un hacker sentándose en otra terminal de programador tenía que gastar la primera hora encontrando qué comandos de macro hacían cada actividad.

Frustado, Steele se dió a la tarea de resolver el problema. El comenzó a recolectar los cuatro distintos paquetes de macros y comenzó a ensamblar una tabla documentando los comandos más útiles. En el curso de la implementación del diseño especificado en la tabla, Steele dice que atrajo la atención de Stallman.

"El comenzó a mirar sobre mi hombro, preguntándome que estaba haciendo", recuerda Steele.

Para Steele, un programador de voz suave que interactuaba con Stallman poco frecuentemente, el recuerdo todavía lo acompaña. Mirar sobre el hombro de otro programador mientra este programaba era una actividad común en el laboratorio de IA. Stallman, el mantenedor de TECO en el laboratorio, le pareció el trabajo de Steele "interesante" y prontamente se dió a la tarea de terminarlo.

"Como me gusta decirlo, yo hice el primer 0.001 por ciento de la implementación y Stallman hizo el resto", dice Steele en medio de una risa.

El nombre del nuevo proyecto, Emacs, vino cortesía de Stallman. Abreviación de "macros de edición" (en inglés "editing macros"), significó la trascendencia evolutiva que había tomado lugar durante la explosión de las macros dos años antes. También tomó ventaja de un hueco en el léxico de la programación de software. Notando una ausencia de programas en ITS que comenzaron con la letra

"E", Stallman escogió Emacs, haciendo posible referenciar al programa con una sola letra. Una vez más, el deseo del *hacker* de la eficiencia, había dejado su marca. [10]

En el transcurso de desarrollar un sistema estándar de comandos de macros, Stallman y Steele tuvieron que atravesar una cuerda floja política. En crear un programa estándar, Stallman estaba en clara violación del principio *hacker* fundamental de "promoción de la descentralización". El también estaba amenazando de cortar la misma flexibilidad que había alimentado la innovación explosiva de TECO en primer lugar.

"De un lado, estabamos tratando de tener un conjunto uniforme de comandos de nuevo; pero de otro lado, queríamos mantenerlo abierto, porque la programabilidad era importante", recuerda Steele.

Para resolver el problema, Stallman, Steele y sus compañeros *hackers* David Moon y David Weinreib limitaron el esfuerzo de estandarización a los comandos WYSIWYG que controlaban como aparecía el texto en la pantalla. El resto del esfuerzo en Emacs sería dedicado a retener la extensiblidad del programa.

Stallamn se enfrentaba ahora a otro dilema: si los usuarios hacian cambios pero no comunicaban dichos cambios al resto de la comunidad, el efecto de la Torre de Babel volvería a emerger en algunos sitios. Recayendo en la doctrina *hacker* de compartir la innovación, Stallman incrustó una frase dentro del código fuente que fijaba los términos de uso. Los usuarios eran libres de modificar y redistribuir el código con la condición de que devolvieran las extensiones que ellos hicieran. Stallman lo llamó la "comuna de Emacs". Justo como TECO se había convertido en algo más que un simple editro, Emacs se había convertido en algo más que un simple programa. Para Stallman, era un contrato social. En un temprano memo en el que documentaba el proyecto, Stallman escribió los términos de tal contrato. "EMACS", el escribión, "fue distribuido con base en la cooperación comunal, que significa que todas las mejoras debían ser devueltas a mi para ser incorporadas y distribuidas". [11]

No todos aceptaban el contrato. La innovación explosiva continuó durante toda la década, resultando en un grán número de programas similares a Emacs con diferentes grados de compatibilidad. Unos cuantos citaban su relación con el Emacs original de Stallman con nombres graciosos y recursivos: Sine (Sine no es Emacs), Eine (Eine no es Emacs), y Zwei (Zwei fue Eine inicialmente). Como un dedicado exponente de la ética *hacker*, Stallman no vió razones para frenar la innovación a través de acciones legales. Sin embargo, el hecho de que algunas personas tomaran tan ansiosamente el software del pecho de la comunidad, lo alteraran, y le colocaran un nuevo nombre en el software resultante mostraba una increíble falta de cortesía.

Tal compartimiento descortés se reflejó contra otros, inquietando a los desarrollos en la comunidad *hacker*. La decisión de Brian Reid en 1979 de incrustar "*bombas de tiempo*" en Scribe, haciendo posible que Unilogic limitará el número de accesos no pagados al software fue un mal augurio para Stallman, "El consideraba esto la cosa más Nazi que había visto en su vida", recuerda Reid. A pesar de alcanzar la fama en Internet, como el creador de la jerarquía *alt* en Usenet, Reid dice que todavía debe vivir con la decisión que tomó en 1979, por lo menos ante los ojos de Stallman. "El dijo que todo el software debería ser libre y que el prospecto de cobrar dinero por el software era un crimen contra la humanidad". [12]

A pesar de que Stallman fue impotente para oponerse a la venta de Reid, el tenía la habilidad para restringir otras formas de comportamiento juzgadas contrarias al *ethos* del *hacker*. Como mantenedor central del código para la "comuna" de Emacs, Stallman comenzó a esgrimir su poder en favor de efectos políticos. Durante las fases finales del conflicto acerca de los sistemas de palabras claves, con los administradores en el Laboratorio para la Ciencia del Computador, Stallman inició una huelga de "software" [13] en la que se negaba a enviar la última versión de Emacs a los miembros del laboratorio

hasta que ellos rechazaran el sistema de seguridad en los computadores del laboratorio. La acción no hizo mucho por mejorar la creciente reputación de extremista de Stallman, pero hizo ver su punto de vista: esperaba que los miembros de la comuna hablaran de los valores básicos del *hacker*.

"Una gran cantidad de personas estaban furiosas conmigo, diciendo que yo estaba tratando de hacerlos rehenes o de chantajearlos, lo cual en algún sentido era lo que yo hacía", diría posteriormente Stallman al autor Steven Levy". "Yo estaba entablando violencia con ellos porque yo pensaba que ellos estaban entablando violencia con todos los demás". [14]

Con el tiempo, Emacs se volvió un a herramienta de venta la ética *hacker*. La flexibilidad que Stallman le había dado al software no solo alentaba a la colaboración, la demandaba. Los usuarios que no querían quedar relegados de los últimos desarrollos en la evolución de Emacs o no querían contribuir sus contribuciones de vuelta a Stallman corrían el riesgo de perderse los últimos cambios. Y los cambios fueron muchos. Veinte años después, los usuarios han modificado a Emacs para tantos usos como lo son hojas de cálculo, calculadoras, bases de datos y navegador web, que los desarrolladores más recientes de Emacs adoptaron una *overflowing sink* para representar su funcionalidad versátil. "Esta era la idea en que queríamos convenir", dijo Stallman. "La cantidad de cosas que contenía es al mismo tiempo maravillosa y terrible".

Los contemporáneos del laboratorio de IA era más caritativos. Hal Abelson, un estudiante graduao del MIT que trabajó con Stallman durante 1970 y que luego asistiría Stallman como miembro de la mesa directiva de la Fundación para el Software Libre, describe a Emacs como "una creación absolutamente brillante". Dando a los programadores la forma de añadir nuevas características y librerías de software sin entrometerse con el sistema, dice Abelson, Stallman pavimentó la vía para futuros proyectos de software de gran escala colaborativos. "Su estructura era lo suficiente robusta como para tener gente colaborando independientemente alrededor del mundo [y] contribuyéndole", dice Abelson. "No se si eso se había hecho antes". [15]

Guy Steele expresa una adminiración similar. Actualmente es un científico investigador para *Sun Microsystem*, el recuerda a Stallman principalmente como un "programador brillante con la habilidad de generar grandes cantidades de código relativamente sin errores". Aunque sus personalidades no concuerdan, Steele y Stallman colaboraron lo suficiente para que Steele echara un vistazo al intenso estilo de Stallman para codificar. El recuerda un episodio notable a finales de la década de 1970 cuando los dos programadores colaboraron para escribir la característica de "impresión bonita" del editor. Originalmente concebida por Steele, la impresión bonita fue una característica activada por una combinación de teclas, que reformateaba el código visto en Emacs de tal forma que fuera al mismo tiempo más legible y tomara menos espacio, avanzando aún más las características WYSIWYG del programa. La característica fue lo suficientemente estratégica para atraer el interés activo de Stallman, y no mucho después de que Steele la escribió, el y Stallman estaban planeando una versión mejorada.

"Nos sentamos una mañana", recuerda Steele. "Yo estaba en el teclado, y el estaba en mi hombro", dice Steele. "El tenía la toda la voluntad de dejarme escribir, pero el me dijo que debía escribir".

La sesión de programación duró 10 horas. Durante todo ese tiempo, dice Steele, ni Stallman tomó un descaso, ni hizó una pequeña caminata. Al final de la sesión, ellos habían logrado escribir el *hack* de la impresión bonita en menos de 100 líneas. "Mis dedos estuvieron en el teclado todo el tiempo", recuerda Steele, "pero se sentía como si nuestras ideas están flotando dentro de la pantalla. El me decía que debía escribir, y yo lo escribía".

La longitud de la sesión se reveló a sí misma cuando Steele finalmente dejo el laboratorio de IA. Parado afuera del edificio en la 545 *Tech Square*, el estaba sorprendido de encontrarse a sí mismo rodeado de la oscuridad noctura. Como programador, Steele estaba acostumbrado a sesiones de codificación maratónicas. Sin embargo, algo de la sesión fue diferente. Trabajando con Stallman había forzado a Steele a bloquear todos los estímulos externos y a concentrar sus energías mentales enteras en la tarea que estaba realizando. Recordando, Steele dice que el ritmo mental de Stallman era al mismo tiempo alegre y miedoso. "Mi primer pensamiento después fue: Ha sido una gran experiencia, muy intensa, y no quiero volver a tenerla en mi vida"

Notas

[1] Ver Josh McHugh, "For the Love of Hacking," [Forbes] (Agosto 10, 1998). http://www.forbes.com/forbes/1998/0810/6203094a.html

[2] See Stallman (1986).

[3] Ver Joseph Weizenbaum, [Computer Power and Human Reason: From Judgment to Calculation] (W. H. Freeman, 1976): 116.

[4] De acuerdo con el archivo de Jargon (*Jargon File*), el nombre original de TECO provino de Editor de Cinta y Corrector (en inglés *Tape Editor and Corrector*). Ver http://www.tuxedo.org/~esr/jargon/html/entry/TECO.html

[5] Ver Richard Stallman, "EMACS: The Extensible, Customizable, Display Editor" Memo del laboratorio de IA (1979). Una versión actualizada en HTML de este memo, del cual estoy citado, está disponible en http://www.gnu.org/software/emacs/emacs-paper.html

[6] Ver Richard Stallman, "Emacs: el editor de pantalla completa" (1987). http://www.lysator.liu.se/history/garb/txt/87-1-emacs.txt

[7] Ver Richard Stallman, "Emacs the Full Screen Editor" (1987). http://www.lysator.liu.se/history/garb/txt/87-1-emacs.txt

[8] Ver Richard Stallman, "Emacs the Full Screen Editor" (1987). http://www.lysator.liu.se/history/garb/txt/87-1-emacs.txt

[9] Ver Richard Stallman, "Emacs the Full Screen Editor" (1987). http://www.lysator.liu.se/history/garb/txt/87-1-emacs.txt

[10] Ver Richard Stallman, "Emacs the Full Screen Editor" (1987). http://www.lysator.liu.se/history/garb/txt/87-1-emacs.txt

[11] Ver Richard Stallman, "EMACS: El extendible y configurable editor de despliegue" Memo del laboratorio de IA (1979). Una versión actualizada en HTML de este memo, del cual estoy citado, está disponible en http://www.gnu.org/software/emacs/emacs-paper.html

[12] En una entrevista de 1996 con el magazín en línea [MEME], Stallman citó la venta de Scribe como molesta, pero dudó en mencionar a Reid por su nombre. "El problema fue que nadie censuró o castigó al estudiante por lo que hizo", dijo Stallman, "El resultado fue que alguna gente fue tentada para seguir su ejemplo". Ver [MEME] 2.04. http://memex.org/meme2-04.html

[13] Ver Steven Levy, [Hackers](Penguin USA [paperback], 1984): 419.

[14] See Steven Levy, [Hackers] (Penguin USA [paperback], 1984): 419.

[15] Durante la escritura de este capítulo, he elegido enfocarme más en la significancia social de Emacs, que en su significancia como software. Para leer más acerca del lado de software, recomiendo el memo de 1979 de Stallman. Particularmente recomiedo la sección titulada "Investigación a través del desarrollo de herramientas instaladas" (#SEC27). No sólo es accesible para el lector no técnico, también arroja luces de que tan cercanas es la filosofía política de Stallman con su filosofía de desarrollo de software. Una pequeña parte dice:

> EMACS no pudo haber sido alcanzado por un proceso de diseño cuidadoso, porque tales procesos llegan sólo a metas que son visibles en el *outset*, y cuya deseabilidad está establecida en la fase inicial del *outset*. Ni siquiera yo o cualquier otro hubiera visualizado un editor extensible hasta que yo hice uno, ni hubiera apreciado su valor hasta que lo hubiera experimentado. EMACS existe porque yo me sentí libre de hacer pequeñas mejoras en un camino cuyo final no estaba a la vista.

Capítulo 7. Una Decisión Moral Difícil

En Septiembre 27 de 1983 los programadores de computadores que entraron al grupo de Usenet net.unix-wizards encontraron un mensaje inusual. Publicado en las primeras horas de la madrugada, a las 12:30 a.m. para ser precisos, y firmado por mailto:rms@mit-oz, el asunto del mensaje era conciso pero llamaba la atención. "Nueva implementación de UNIX," decía. Sin embargo, en lugar de presentar una versión de Unix recién lanzada, el párrafo inicial del mensaje enviaba un llamado a tomar las armas:

> Comenzando este Día de Acción de Gracias voy a escribir un sistema compatible con Unix llamado GNU (que significa Gnu No es Unix), y lo voy a entregar gratis para cualquiera que desee utilizarlo. Contribuciones en tiempo, dinero, programas y equipos son bastante necesarias. [1]

Para el desarrollador experimentado de Unix, el mensaje era una mezcla de idealismo y hubris. No solamente pretendía el autor reconstruir el ya maduro sistema operativo Unix empezando de cero, sino que también proponía mejorarlo en algunos lugares. El nuevo sistema GNU, predecía el autor, contendría todos los componentes usuales --un editor de texto, un programa de consola para ejecutar aplicaciones compatibles con Unix, un compilador y "algunas otras cosas." [2] También contendría muchas características atractivas qie otros sistemas Unix no ofrecían aún: una interfaz gráfica de usuario, basada en el lenguaje de programación Lisp, un sistema de archivos a prueba de caidas, y protocolos de red construidos de acuerdo con el sistema de red interno de MIT.

"GNU será capaz de ejecutar programas de Unix, pero no será idéntico a Unix," escribía el autor. "Haremos todas las mejoras que sean convenientes, basados en nuestra experiencia con otros sistemas operativos."

Anticipando una respuesta escéptica de parte de algunos lectores, el autor se aseguró de ubicar a continuación de su descripción del sistema operativo una breve referencia biográfica titulada "¿Quién soy yo?":

> Yo soy Richard Stallman, inventor del original y muy imitado editor EMACS, ahora en el laboratorio de Inteligencia Artificial en MIT. He trabajado extensivamente en compiladores, editores, depuradores, intérpretes de comandos, el Sistema Incompatible de Tiempo Compartido (Incompatible Timesharing System ó ITS) y el sistema operativo de la Máquina Lisp. Inicié el soporte de visualización independiente de la terminal en el ITS. Además he implementado un sistema a prueba de caídas y dos sistemas de ventanas para máquinas Lisp. [3]

Por asuntos del destino, el elegante Proyecto GNU de Stallman no pudo cumplir su fecha de lanzamiento de Día de Acción de Gracias. Para enero de 1984, sin embargo, Stallman cumplió su promesa y se involucró plenamente en el mundo del desarrollo de software de Unix. Para un arquitecto de software criado en ITS, era como diseñar centros comerciales de los suburbios en vez de palacios Moros. Aún así, construir un sistema operativo parecido a Unix tenía sus ventajas ocultas. ITS había sido poderoso, pero también tenía un talón de Aquiles: Los hackers de MIT lo habían diseñado específicamente para aprovechar la línea de máquinas PDP, creada por DEC. Cuando los administradores del laboratorio de IA decidieron sacar de circulación la poderosa máquina PDP-10 del laboratorio a comienzos de los ochentas, el sistema operativo que los hackers habían asimilado a una

vibrante ciudad se convirtió instantánea,emte en un pueblo fantasma. Unix, por otro lado, estaba diseñado para movilidad y supervivencia a largo plazo. Inicialmente desarrollado por científicos menores en AT&T, el programa se había escapado del radar de la gerencia corporativa, encontrando un hogar feliz en el mundo presupuestalmente limitado de los sistemas de computadores académicos. Con menos recursos que sus colegas en MIT, los desarrolladores de Unix habían adaptado el software para funcionar en un conjunto muy disímil de sistemas de hardware: todo desde el pequeño PDP-11-a de 16 bits, considerado útil solamente para tareas pequeñas por la mayoría de los hackers del Laboratorio de IA, hasta *mainframes* de 32 bits como el VAX 11/780. Para 1983 unas pocas compañías, notablemente Sun Microsystems, iban aún mas lejos, desarrollándo una nueva generación de microcomputadores, llamados "estaciones de trabajo," para aprovechar el cada vez mas ubicuo sistema operativo.

Para facilitar este proceso, los desarrolladores encargados de diseñar las familias dominantes de Unix se aseguraban de dejar una capa adicional de abstracción entre el *software* y la máquina. En lugar de adaptar el sistema para aprovechar los recursos de una máquina específica -- como habían hecho los hackers del Laboratorio de IA con ITS y PDP-10 -- los desarrolladores de Unix preferían una aproximación más genérica. Concentrándose más en los estándares y especificaciones de interconexión que mantenían los múltiples componentes de un sistema operativo juntos que en los componentes mismos, crearon un sistema que podía ser rápidamente modificado para adaptarse a los gustos de cualquier máquina. Si un usuario se alteraba alguna parte, los estándares hacían posible retirar un subcomponente individual y arreglarlo o reemplazarlo con algo mejor. De manera sencilla, lo que le faltaba a la aproximación de Unix en estilo a estética quedaba más que compensado en términos de flexibilidad y economía, de ahí su rápida adopción. [4]

La decisión de Stallman de comenzar a desarrollar el sistema GNU fue motivada por el final del sistema ITS en el que los hackers del Laboratorio de AI habían trabajado por tanto tiempo. El final del sistema ITS fue un golpe traumático para Stallman. Al llegar poco después del episodio de la impresora laser de Xerox, ofrecía aún mas evidencias de que la cultura hacker del laboratorio de IA estaba dejando de ser inmune a las prácticas de negocios del mundo exterior.

Como el código del software que lo componía, las raíces del final de ITS se extendían muy atrás. El gasto de defensa, por mucho tiempo una fuente importante para el desarrollo de las ciencias de la computación, se había terminado durante los años posteriores a Vietnam. En una lucha desesperada por nuevos fondos los laboratorios y universidades empezaron a buscar en el sector privado. En el caso del Laboratorio de IA ganarse los inversionistas privados fue fácil. Hogar de algunos de los proyectos más ambiciosos de ciencias de computación de la posguerra, el laboratorio se convirtió en un incubador rápido de tecnología. De hecho, para 1980, la mayoría del personal del laboratorio, incluyendo muchos hackers, dividían su tiempo entre los proyectos del instituto y proyectos comerciales.

Lo que inicialmente parecía como un trato de ganancia-ganancia -los hackers trabajaban en los mejores proyectos, dándole al laboratorio la posibilidad de ser el primero en tener acceso a las más novedosas tecnologías de computadores - pronto se reveló como un trato Faustiano. Mientras más tiempo dedicaban los hackers a los proyectos comerciales de punta, menos tiempo tenían para dedicarle al mantenimiento de la infraestructura barroca de software del laboratorio. Pronto las compañías comenzaron a contratar hackers directamente en un intento de monopolizar su tiempo y atención. Con menos hackers para preocuparse del taller, los programas y las máquinas tomaban mas tiempo en ser arregladas. Aún peor; según Stallman, el laboratorio comenzó a sufrir un "cambio demográfico". Los hackers que alguna vez habían formado una minoría vocal dentro del Laboratorio de Inteligencia Artificial estaban perdiendo su membresía a la vez que "los profesores y estudiantes que no querían realmente el [PDP-10] eran tan numerosos como antes." [5]

El punto de corte vino en 1982. Ese fue el año en que la administración del laboratorio decidió actualizar su computador principal, el PDP-10. Digital, la compañía que fabricaba el PDP-10 había descontinuado la línea. A pesar de que la compañía aún ofrecía un *mainframe* de alto poder, llamado el KL-10, la nueva máquina requería una reescritura drástica o "port" de ITS si los hackers querían continuar corriendo con el mismo sistema operativo. Temiendo que el laboratorio hubiera perdido su masa crítica de programadores talentosos, los miembros de la facultad del laboratorio presionaron para adoptar Twenex, un sistema operativo desarrollado por Digital. Superados en número los hackers no tuvieron otra opción que aceptar.

"Sin hackers para mantener el sistema, decían [los miembros de la facultad], ‘Vamos a tener un desastre; debemos tener software comercial,’" Stallman recordaría algunos años después. "Ellos dijeron, ‘Podemos esperar que la compañía lo mantenga.’ Se comprobó que estaban completamente equivocados, pero eso fue lo que hicieron." [6]

Al principio, los hackers vieron el sistema Twenex como otro símbolo autoritario esperando ser derrotado. El mismo nombre del sistema era una protesta. Oficialmente llamado TOPS-20 por DEC, era un sucesor de TOPS-10, un sistema operativo comercial que DEC mercadeaba para el PDP-10. Bolt Beranek Newman había desarrollado una versión mejorada, llamada Tenes, en la que TOPS-20 se basó. [7] Stallman, el hacker que acuñó el término Twenex, dice que pensó en el nombre como una manera de evitar usar el nombre TOPS-20. "El sistema no era lo mejor (*tops*, en inglés) por lo que de ninguna manera lo iba a llamar así," recuerda Stallman. "Por lo que decidí incluir una ’w’ en el nombre Tenex y llamarlo Twenex."

La máquina que corría el sistema Twenex/TOPS-20 tenía su propio sobrenombre burlón: Oz. De acuerdo a una leyenda hacker, la máquina obtuvo su alias porqué requería una máquina PDP-11 más pequeña para accionar su terminal. Un hacker, al ver la configuración del KL-10-PDP-11 por primera vez lo vinculó a la espectacular aparición en pantalla del Mago de Oz. "Yo soy en grande y poderoso Oz," entonaba el hacker. "No le presten atención al PDP-11 detrás de esa consola." [8]

Si los hackers se rieron la primera vez que se encontraron con el KL-10, su risa se desvaneció rápidamente cuando se encontraron con Twenex. No solamente tenía seguridad incluida, sino que los ingenieros de software habían diseñado las herramientas y aplicaciones con el sistema de seguridad en mente. Lo que alguna vez había sido un juego de gato y ratón sobre los paswords, en el caso del sistema de seguridad del Laboratorio de Ciencias de la Computación, se convirtió en una batalla a campo abierto sobre el manejo del sistema. Los administradores del Systema argumentaron que, sin seguridad, el sistema Oz era más vulnerable a caídas accidentales. Los hackers replicaron que las caídas podían prevenirse mejor atacando el código fuente. Desafortunadamente, el número de hackers con el tiempo y la inclinación para desarrollar este tipo de ataque se había reducido hasta el punto que el argumento del administrador del sistema prevaleció.

Regalando sus claves de acceso y deliberadamente haciendo caer el sistema para recolectar evidencia del desastre resultante, Stallman logró derrotar el intento de obtener control por parte del administrador del sistema. Después de una de estas "tomas de poder" fallidas, Stallman envió una alerta a todo el personal de IA. [9]

"Ha habido otro intento de obtener del poder," escribió Stallman. "Hasta ahora, las fuerzas aristocráticas han sido derrotadas." Para proteger su identidad, Stallman firmó el mensaje "Radio Libre OZ."

El disfraz fue, en el mejor de los casos, muy débil. Para 1982 la aversión de Stallman hacia las claves de acceso y el secreto era tan conocida que usuarios fuera del laboratorio de IA estaban usando su cuenta como punto de entrada para ARPAnet, la red de computadores financiada con dineros para investigación que serviría como base para lo que hoy es internet. Uno de aquellos "turistas" a comienzos de los 80 era Don Hopkins, un programador de California que se enteró por la comunidad hacker de que todo lo que un extraño debía hacer para obtener acceso al preciado sistema ITS de MIT era ingresar con las iniciales RMS e incluir el mismo monograma de tres letras cuando el sistema solicitará una clave de acceso.

"Yo estaré eternamente agradecido de que MIT me dejara a mi y a muchos otros utilizar sus computadores gratis." dice Hopkins. "Significó mucho para mucha gente."

Esta política de "turistas", que había sido abiertamente tolerada por la gerencia de MIT durante los años del ITS, [10] fue dejada a un lado cuando Oz se convirtió en el vínculo principal del laboratorio con ARPAnet. Al principio, Stallman continuó con su política de repetir su nombre de usuario como clave de acceso, para que los usuarios externos pudieran seguir sus pasos. Con el tiempo, sin embargo, la fragilidad de Oz obligó a contener a los usuarios externos que podrían, por accidente o por albergar intenciones malignas, hacer caer el sistema. Cuando esos mismos administradores obligaron a Stallman a dejar de publicar su clave de acceso, Stallman, alegando ética personal se negó a hacerlo y dejó de utilizar el sistema Oz del todo. [11]

"[Cuando] las claves de acceso aparecieron por primera vez en el laboratorio de IA de MIT yo [decidí] seguir mi creencia de que no deben haber claves de acceso," Stallman diría mas tarde. "Como yo no creo que sea realmente deseable que haya seguridad en un computador, no debería estar dispuesto a ayudar a mantener un régimen de seguridad." [12]

La renuencia de Stallman a inclinarse ante el grande y poderoso Oz simbolizaba la creciente tensión entre los hackers y la gerencia del laboratorio de IA a comienzos de los 80. Esta tensión no era nada comparada con el conflicto que existía en el seno mismo de la comunidad hacker. Para el momento en que llegó el KL-10 la comunidad hacker ya se había dividido en dos campos. El primero se movía alrededor de una compañía de software llaman Symbolics, Inc. El segundo se movía en torno a el principal rival de Symbolics, Lisp Machines, Inc. (LMI). Ambas compañías se encontraban en una carrera para llevar al mercado la Máquina Lisp, un dispositivo construido para aprovechar plenamente del Lenguaje de Programación Lisp.

Creado por el pionero en la investigación de la inteligencia artificial, John McCarthy, investigador en inteligencia artificial de MIT, a finales de los años 50, Lisp es un lenguaje elegante, muy útil para programas encargados de procesamiento y ordenamiento en grandes cantidades. El nombre del lenguaje es una versión reducida de Procesamiento de Listas (*LISt Processing*). Tras la salida del McCarthy hacía el Laboratorio de Inteligencia Artificial de Stanford, los hackers de MIT refinaron el lenguaje en un dialecto local llamado MACLIST. El "MAC" se refería al Proyecto MAC, un proyecto de investigación financiado por DARPA que dió origen al laboratorio de IA y al Laboratorio para las Ciencias de la Computación. Dirigidos por el arquitecto-hacker Richard Greenblatt, los programadores del laboratorio de AI contruyeron durante los 70 un sistema operativo completo basado en Lisp, llamado el sistema operativo de la Máquina Lisp (*Lisp Machine Operating System*). Para 1980, el proyecto de la Máquina Lisp había generado dos iniciativas comerciales. Symbolics, encabezado por Russell Noftsker, un antiguo administrador del laboratorio de IA, y Lisp Machines, Inc.,encabezado por Greenblatt.

El software de la Máquina Lisp era construido por hackers, es decir, era propiedad de MIT pero estabaa disponible para que cualquiera lo copiara, como una costumbre de hackers. Ese sistema limitaba las ventajas de mercadeo de cualquier compañía que esperara licenciar el software a MIT y mercadearlo como único. Para asegurarse una ventaja, y para favorecer los aspectos que los clientes podrían considerar atractivos, las compañías reclutaron a algunos de los hackers del Laboratorio de IA y los pusieron a trabajar en varios componentes del sistema operativo de la Máquina Lisp fuera de la tutela del Laboratorio de IA.

La compañía que aplicaba esta estrategia de manera más agresiva era Symbolics. Para finales de 1980 había contratado 14 de los miembros del equipo de AI del laboratorio como consultores de medio tiempo para desarrollar su versión de la Máquina Lisp. Con excepción de Stallman, los demás firmaron para ayudar a LMI. [13]

En un principo, Stallman aceptó el intento de ambas compañías de comercializar la máquina Lisp, a pesar de que significaba más trabajo para el. Ambas licenciaron el código fuente del sistema operativo de la Máquina Lisp a MIT, y el trabajo de Stallman era actualizar la Máquina Lisp del laboratorio, para mantenerse al día con las últimas innovaciones. A pesar de que la licencia de Symbolics con MIT le daba a Stallman el derecho a revisar, pero no copiar el código fuente de Symbolics, Stallman dice que un "acuerdo de caballeros" entre la gerencia de Symbolics y el Laboratorio de IA hacía posible utilizar porciones atractivas del código a la manera hacker tradicional.

En marzo 16 de 1982, una fecha que Stallman recuerda bien por ser su cumpleaños, los ejecutivos de Symbolics decidieron terminaron con este acuerdo de caballeros. La jugada fue en gran parte un movimiento estratégico. LMI, la competencia más importante en el mercado de la Máquina Lisp estaba esencialmente usando una copia de la máquina Lisp del Laboratorio de IA. En lugar de subsidiar el desarrollo de un rival de mercado, los ejecutivos de Symbolics eligieron hacer cumplir los puntos de la licencia. Si el Laboratorio de AI quería que su sistema operativo se mantuviero actualizado con el de Symbolics, el laboratorio debería cambiarse a una máquina Symbolics y acabar con sus vínculos con LMI.

Como la persona responsable de mantener la Máquina Lisp del laboratorio Stallman estaba indignado. Viendo este anuncio como un "ultimatum," Stallman respondió desconectando el canal de comunicación de microondas entre Symbolics y el laboratorio. Luego hizo votos para nunca trabajar en una máquina Symbolics y se alió inmediatamente con LMI. "De la forma en yo lo veía, el Laboratorio de IA era un país neutral, como Bélgica en la Segunda Guerra Mundial," dice Stallman. "Si Alemania invade Bélgica, Bélgica le declara la guerra a Alemania y se alía con Gran Bretaña y Francia."

Las circunstancias de la llamada "Guerra de Symbolics" de 1982-1983 dependen fuertemente de quién sea el que cuente la historia. Cuando los ejecutivos de Symbolics se dieron cuenta de que sus características mas novedosas aún aparecían en la Máquina Lisp del Laboratorio de IA y, por extensión en la máquina de LMI, instalaron un programa "espía" en la terminal de computador de Stallman. Stallman dice que estaba reescribiendo las nuevas características de ceros, aprovechando una claúsula de la licencia que permitía las revisiones, pero también incurriendo en esfuerzos para que el código fuente fuera bastante diferente. Los ejecutivos de Symbolics no estuvieron de acuerdo y llevaron su caso ante la administración de MIT. De acuerdo al libro de 1994, [The Brain Makers: Genius, Ego, and Greed, and the Quest for Machines That Think], escrito por Harvey Newquist, la administración respondió con una advertencia a Stallman para que se "mantuviera alejado" del proyecto de la Máquina Lisp. [14] Según Stallman los administradores de MIT lo respaldaron. "Nunca fui amenzado," dice. "Si hice algunos cambios en mi forma de hacer las cosas, sin embargo. Sólo para estar ultra seguro, ya no leía el código fuente. Usaba solamente la documentación y después escribía el

código a partir de eso."

Sin importar el resultado, la discusión solidificó la resolución de Stallman. Sin código fuente para revisar, Stallman llenó los vacíos en el software de acuerdo a sus propios gustos y enlistó a los miembros del Laboratorio de IA para que le produjeran un flujo continuo de reportes de error. También se aseguró de que los programadores de LMI tuvieran acceso directo a los cambios. "Iba a castigar Symbolics aunque fuera lo último que hiciera," dice Stallman.

Este tipo de declaraciones son reveladoras. No solo arrojan una luz sobre la naturaleza no pacifista de Stallman, sino que también reflejan el intenso nivel de emoción provocado por el conflicto. De acuerdo a otra historia relacionada con Newquist, Stallman estaba tan enfurecido en un punto que escribió un correo electrónico amenazando "envolverme a mi mismo en dinamita y entrar en las oficinas de Symbolics." [15] A pesar de que Stallman niega recordar el correo electrónico, y aún describe su existencia como un "rumor vicioso," el es conciente de que esos pensamientos si entraron a su cabeza. "Yo definitivamente si tuve fantasías de matarme a mi mismo y destruir su edificio en el proceso," dice Stallman. "Pensé que mi vida había terminado." [16]

Este nivel de frustración era debido en gran parte a lo que Stallman veía como la "destrucción" de su "hogar" --i.e., el final de la subcultura hacker del Laboratorio de IA. En una entrevista posterior por correo electrónico con Levy, Stallman se vincularía a si mismo con la figura histórica de Ishi, el último miembro sobreviviente de los Yahi, una tribú del Nordeste de la costa Pacífica aniquilada durante las Guerras Indias de las décadas de 1860 y 1870. La analogía ubica la supervivencia de Stallman en términos épicos, casi míticos. En la realidad, sin embargo, habla un poco más acerca de la tensión entre Stallman y sus compañeros hackers del laboratorio de IA antes del asunto Symbolics-LMI. En lugar de ver a Symbolics como una fuerza exterminadora, muchos de los colegas de Stallman la veían como una posibilidad tardía de relevancia. Comercializando la Máquina Lisp, la compañía expulsó los conceptos de diseño de software manejado por ingenieros de los confines de la torre de márfil del laboratorio de IA hacía el mercado, donde mandaban los principios de diseño de los gerentes. En lugar de ver a Stallman como un adalid, muchos hackers lo veían como un anacronismo problemático.

Stallman no disputa esta visión alternativa de los eventos históricos. De hecho, el dice que fue una razón más para aumentar la hostilidad causada por el "ultimatum" de Symbolics. Incluso antes de que Symbolics contratara a la mayoría del personal hacker del laboratorio de IA, Stallman dice que muchos de los hacker que más tarde se unirían a Symbolics lo estaban evitando. "Ya no estaba siendo invitado a Chinatown," recierda Stallman. "La costumbre iniciada por Greenblatt era que si uno salía a comer, iba de un lugar a otro o enviaba un mensaje preguntando a cualquiera en el laboratorio si deseaba ir. El algún momento entre 1980 y 1981 dejaron de preguntarme. No sólo no estaba siendo invitado, sino que una persona me confesó mas tarde que había sido presionado para mentirme con el fin de mantener una salida a comer sin mi como un secreto."

A pesar de que Stallman sentía furia hacía aquellos hackers que orquestaron esta singular forma de ostracismo, la controversia de Symbolics generó un nuevo tipo de furia, la furia de una persona a punto de perder su hogar. Cuando Symbolics dejó de enviar el código fuente de sus cambios, Stallman respondió encerrándose en sus oficinas de MIT y reescribiendo cada nueva característica y herramienta desde ceros. Tan frustrante como puede haber sido, garantizaba que los usuarios futuros de la Máquina Lisp tuvieran acceso irrestricto a las mismas características que los usuarios de Symbolics.

También garantizó la condición legendaria de Stallman dentro de la comunidad hacker. Ya conocido por su trabajo con Emacs, la habilidad de Stallman para igualar los resultados de todo un equipo de programadores de Symbolics --un equipo que, por sí mismo, incluía a mas de un hacker legendario--

aún es considerado como uno de los mayores logros humanos de la Era de la Información, o de cualquier era. Llamandolo un "*hack* maestro" y a Stallman mismo un "John Henry virtual del código de computador, " el autor Steven Levy anota que muchos de sus rivales empleados por Symbolics no tenían otra opción que respetar, a pesar suyo, a su antiguo camarada idealista. Levy se refiere a las declaraciones de Bill Gosper, un hacker que eventualmente se fue a trabajar para Symbolics en la oficina de la compañía en Palo Alto, cuando expresa su asombro ante los resultados de Stallman durante este periodo:

> Podía ver algo que Stallman escribió, y podía decidir que era malo (probablemente no, pero alguién podría convencerme de que era malo), y yo todavía diría "Pero, esperen un momento --Stallman no tiene nadie con quién discutir toda la noche allá. ¡Esta trabajando solo! ¡Es increible que alguién pudiera hacer esto solo!" [17]

Para Stallman los meses gastados jugando a igualar a Symbolics evocan una mezcla de orgullo y profunda tristeza. Como un liberal cuyo padre había participado en la Segunda Guerra Mundial Stallman no es ningún pacifista. De muchas maneras la guerra contra Symbolics ofreció el rito de paso hacía el que Stallman se había ido acercando desde que se había unido al personal del Laboratorio de IA una década antes. Al mismo tiempo, sin embargo, coincidió con la destrucción traumática de la cultura hacker que había acogido a Stallman desde sus años de adolescencia. Un día, mientras tomaba un descanso de escribir código, Stallman experimentó un momento traumático cuando pasaba por la sala de equipos de laboratorio. Allí Stallman encontró el aparatoso armazón de la máquina PDP-10, ya en desuso. Impresionado por las luces apagadas, luces que alguna vez se encendieron activamente para indicar el estado del programa interno, Stallman dice que el impacto emocional no fue muy diferente de aquel que se siente al encontrarse con el cadáver bien conservado de algún miembro de la familia muy querido.

"Empecé a llorar allí mismo en la sala de equipos, " dice. "Viendo la máquina allí, muerta, con nadie más para arreglarla vi como mi comunidad había sido completamente destruida."

Stallman tendría muy pocas oportunidades para lamentarse. A pesar de todo el furor que había causado y de todo el trabajo necesario para desarrollarla, no era más que un espectáculo de segunda en las grandes batallas del mercado de la tecnología. El incansable ritmo de la miniaturización estaba trayendo nuevos y más poderosos microprocesadores que pronto de incoroporarían en las capacidades de *hardware* y *software* de la misma manera en que una ciudad moderna se traga un pequeño poblado desierto.

En la cresta de esta ola de microprocesadors se encontraban cientos de miles de programas de software comercial, cada uno protegido por un intrincado conjunto de licencias y acuerdos de privacidad que hacía imposible para los hackers revisar o compartir el código fuente. Las licencias eran toscas y a veces no encajaban muy bien, pero para 1983 ya eran lo suficientemente fuertes como para satisfacer a las cortes y asustar a quienes quisiseran transgredirlas. El *software*, alguna vez una forma de aderezo que la mayoría de las compañías de hardware regalaban para darle a sus sistemas más sabor, se estaba convirtiendo rápidamente en el plato principal. En su creciente hambre por nuevos juegos y características los usuarios estaban dejando de lado la solicitud tradicional de revisar la receta después de cada comida.

En ninguna parte era este estado de las cosas más evidente que en el campo de los sistemas personales de computadores. Compañías como Apple Compter y Commodore estaban creando nuevos millonarios vendiendo máquinas con sistemas operativos incluidos. Sin saber de la existencia de la cultura hacker y de su disgusto por el software que se distribuía solemente en binario, la mayoría de los usuarios no vieron necesidad de protestar cuando estas compañías no incluyeron los archivos de

código fuente. Algunos anárquicos seguidores de la ética hacker ayudaron a promover esa ética dentro de este nuevo mercado, pero en su mayor parte el mercado recompensaba a sus programadores lo suficientemente rápido como para que escribieran nuevos programas y con el suficiente sentido común como para proteger los trabajos legalmente por medio de derechos de autor.

Uno de los mas notables de estos programadores era Bill Gates, un desertor de Harvard dos posterior a Stallman. A pesar de que Stallman no lo sabía en ese momento, siete años antes de enviar su mensaje al grupo de noticias net.unix-wizards Gates, un naciente emprendedor y socio de la firma de software de Albuquerque Micro-Soft, que después se escribiría Microsoft, había enviado su propia carta abierta a la comunidad de desarrolladores. Escrita como respuesta a la actitud de los usuarios de PC de copiar los programas de Micro-Soft, la "Carta abierta a los Aficionados" (*Open Letter to Hobbyists*) había criticado fuertemente la noción de desarrollo comunitario de software.

"¿Quién puede darse el lugo de hacer trabajo profesional por nada?" preguntaba Gates. "¿Que aficionado puede ponerle tres años hombre a programar, corregir y documentar su producto para después distribuirlo gratis?" [18]

A pesar de que solo algunos hackers del Laboratorio de IA vieron la misiva, la carta de Gates de 1976 representaba la cambiante actitud hacía el software tanto entre compañías de software comercial como entre desarrolladores de software comercial. ¿Por qué tratar al software como un bien de costo cero cuando el mercado decía otra cosa? Cuando los años 70 le dieron paso a los 80 vender software se volvió mucho más que una manera de pagar los costos, se convirtió en una declaración política. En un momento en que la Administración Reagan estaba rápidamente desmantelando muchas de las regulaciones federales y programas de gastos que se habían construido durante el medio siglo que siguió a la gran depresión, muchos programadores de software vieron la ética hacker como anticompetitiva y, por extensión, no americana. En el mejor de los casos era un recuerdo de las actitudes anticorporativas de finales de los sesentas y comienzos de los setentas. Como si un banquero descubriera un poncho hippie entre sus camisas y corbatas, muchos programadores de computadores trataban la ética hacker como un vestigio embarazoso de una era idealista.

Como un hombre que había permanecido durante todos los sesentas como un embarazoso retorno a los cincuentas, a Stallman no le importaba vivir fuera de tiempo con sus compañeros. Como un programador acostumbrado a trabajar con las mejores máquinas y el mejor software, sin embargo, Stallman se enfrentaba lo que él solo podía describir como una "difícil decisión moral": o se sobreponía a su objeción hacía el software "propietario" --el término que Stallman y los hackers utilizaban para describir cualquier programa que incluyera derechos de autor privados o una licencia que restringiera la copia y modificación-- o dedicaba su vida a construir un sistema alterno, no propietario, de programas de software. Dado su reciente enfrentamiento de varios meses contra Symbolics, Stallman se sentía mucho mas cómodo con la segunda opción. "Supongo que pude haber dejado de trabajar con software del todo," dice Stallman. "No tenía habilidades especiales, pero estoy seguro de que me pude haber convertido en un mesero. No en un restaurante elegante, probablemente, pero pude haber sido un mesero en alguna parte."

Convertirse en mesero --i.e., abandonar del todo la programación --habría significado renunciar totalmente a una actividad, programación de computadores, que le había proporcionado tanto placer. Mirando hacía atrás su vida desde que se mudó a Cambridge, Stallman encuentra fácil identificar largos periodos cuando la programación de software proporcionó el único placer. En lugar de desertar, Stallman decidió perseverar.

Como ateo, Stallman rechaza nociones como el destino, el dharma o una llamada divina. Sin embargo, él si siente que la decisión de evitar el software propietario y construir un sistema operativo para ayudar a otros a hacer lo mismo fue natural. Después de todo fue la combinación de terquedad, visión y "virtuosismo en la codificación" del mismo Stallman los que lo motivaron a considerar una división en un camino cuya existencia ignoraban muchos. Cuando describe esta decisión en un capítulo para el libro de 1999 [Open Sources], Stallman se refiere al espiritú latente en las palabras de la saga judía de Hillel:

> Si yo no soy para mi mismo, entonces quién será para mi? Si sólo soy para mi mismo entonces, ¿Qué soy? Si no es ahora ¿Cuando? [19]

Cuando habla frente al público, Stallman evita el camino religioso y expresa la decisión en términos pragmáticos. "Me preunté a mi mismo: ¿Qué puedo yo, un desarrollador de sistemas operativos, hacer para mejorar la situación? No fue sino hasta que examiné la pregunta por un tiempo que me di cuenta de que un desarrollador de sistemas operativos era exactamente lo que se necesitaba para resolver el problema."

Una vez llegó a esta decisión, según Stallman, todo lo demás "quedó en su sitio." Se abstendría de usar programas de software que comprometieran sus creencias éticas, y al mismo tiempo dedicaría su vida a crear software para facilitarle a otros seguir el mismo camino. Como parte de su cruzada por construir un sistema operativo con software libre o a "morir intentando; de viejo, por suspuesto," añade Stallman, renunció en MIT en enero de 1984, para construir GNU.

La renuncia distancio el trabajo de Stallman de los auspicios legales de MIT. Sin embargo, Stallman aún tenía suficientes amigos y aliados dentro del Laboratorio de IA como para mantener su oficina en MIT sin pagar arriendo. También tenía la habilidad de asegurarse trabajos de consultoría externos para subvencionar los estadios iniciales de desarrollo del Proyecto GNU. Al renunciar de MIT, sin embargo, Stallman imposibilitó cualquier debate sobre conflicto de interés o propiedad del software por parte del instituto. Stallman, cuyo miedo al aislamiento social lo había inclinado a involucrarse cada vez más con el laboratorio de IA durante su adultez temprana, estaba ahora construyendo una pared legal entre él mismo y ese entorno.

Durante los primeros meses, Stallman operó también aislado de la comunidad Unix. A pesar de que su anuncio en el grupo net.unix-wizards había atraído respuestas de simpatizantes, muy pocos voluntarios se enlistaron para unirse a la cruzada en sus primera etapas.

"La reacción de la comunidad fue en gran parte uniforme," recuerda Rich Morin, líder de un grupo de usuarios de Unix en ese entonces. "La gente decía, ‘Oh, es una gran idea. Muéstrenos su código. Muéstrenos que puede hacerse.’"

A la manera realmente hacker, Stallman comenzó a buscar programas y herramientas existentes que pudieran ser convertidas y programas y herramientas GNU. Uno de los primeros fue un compilador llamado VUCK, que convertía los programas escritos en el popular lenguaje C en código de máquina. Traducido del holandés, el acrónimo quería decir Kit de Compilador de la Universidad Libre (*Free University Compiler Kit*), optimista, Stallman le preguntó al autor del programa si este efectivamente era libre. Cuando este le informó que las palabras "Universidaad Libre" se referían a la *Vrije Universiteit*, en Amsterdam Stallman estaba decepcionado.

"El le contestó sarcásticamente que la Universidad era libre, pero el compilador no," recuerda Stallman. "Yo en consecuencia decidí que mi primer prigrama para el Proyecto GNU sería un compilador multilenguaje y multiplataforma." [20]

Eventualmente Stallman encontró un compilador del lenguage Pastel escrito por programadores del Laboratorio Nacional Lawrence Livermore. De acuerdo a la información que Stallman poseía en ese momento, el compilador era libre para ser copiado y modificado. Desafortunadamente, el programa tenía una falla de diseño importante: grababa cada uno de los programas en la memoria base, por lo que acaparaba espacio, necesario para otras actividades del software. En sistemas tipo *mainframe* esta falla de diseño era perdonable. En sistemas Unix era una barrera inhabilitante, ya que las máquinas que corrían Unix eran demasiado pequeñas para manejar los enormes archivos generados. Stallman hizo progresos sustanciales en un comienzo, construyendo una interfaz compatible con C para el compilador. Para el verano, sin embargo, había llegado a la conclusión de que debería construir un compilador totalmente nuevo desde ceros.

En septiembre de 1984, Stallman archivó el desarrollo de compiladores a corto plazo y comenzó a buscar objetivos más cercanos. Inició el desarrollo de una versión GNU de Emacs, el programa que el mismo había supervisado por una década. La decisión fue estratégica. Dentro de la comunidad Unix, los dos editores nativos eran vi, escrito por el cofundador de Sun Microsystems, Bill Joy y ed, escrito por el científico de los Laboratorios Bell (y co-creador de Unix) Ken Thompson. Ambos eran útiles y populares, pero ninguno de los dos ofrecía la naturaleza infinitamente expandible de Emacs. En el proceso de reescritura de Emacs para los interesados en Unix, Stallman tenía mejores posibilidades de mostrar sus habilidades. Además, tenía sentido pensar que los usuarios de Emacs estarían mas sintonizados con la mentalidad de Stallman.

Mirando hacía atrás, Stallman dice que el no vió la decisión en términos estratégicos. "Quería un Emacs, y tenía una buena oportunidad de desarrollar uno."

Una vez más, la idea de reinventar la rueda irritaba la sensibilidad de hacker eficiente de Stallman. En el proceso de escribir una versión de Emacs para Unix, Stallman estaba pronto siguiendo los pasos del estudiante graduado de Carnegie Mellon James Gosling, autor de una versión de Emacs basada en C llamada Gosling Emacs o GOSMACS. La versión de Gosling incluía un intérprete que aprovechaba un subconjunto simplificado del leguaje LISP llamado MOCKLISP. Decidido a construir GNU Emacs con unos cimientos de LISP similares, Stallman aprovechó muchas de las innovaciones de Gosling. A pesar de que Gosling había protegido a GOSMACS con derechos de autor y había vendido los derechos a UniPress, una compañía de software privada, Stallman se basó en las afirmaciones de un compañero desarrollador que había participado en el desarrollo inicial del intérprete de MOCKLISP. De acuerdo al desarrollador, Gosling, mientras que era un estudiante de Ph.D. en Carnegie Mellon, había asegurado a sus colaboradores iniciales que su trabajo se mantendría accesible. A pesar de esto, cuando UniPress se dió cuenta de la naturaleza del proyecto de Stallman amenazó con hacer válidos los derechos de autor. De nuevo Stallman se enfrentaba a la idea de comenzar todo desde ceros.

En el proceso de ingeniería reversa del intérprete de Gosling, Stallman crearía un intérprete de Lisp completamente funcional, haciendo innecesaria la primera versión del intérprete de Gosling. Sin embargo, la idea de ver a los desarrolladores vendiendo derechos sobre el software --de hecho, la noción misma de los desarrolladores teniendo derechos para vender en primer lugar-- molestaba profundamente a Stallman. En un discurso en el Instituto Técnico Real de Suecia, en 1986, Stallman citó el incidente con UniPress como otro ejemplo de los peligros asociados con el software propietario.

"A veces pienso que posiblemente una de las mejores cosas que podría hacer con mi vida es encontrar una pila enorme de software propietario que sea secreto comercial, y comenzar a regalar copias en una esquina de la calle para que no sea más un secreto comercial," dijó Stallman. "Posiblemente esa sea una manera mucho más eficiente de darle a la gente nuevo software libre, mucho más que escribirlo yo mismo; pero todos son demasiado cobardes como para siquiera aceptarlo." [21]

A pesar de la tensión que generó, la disputa sobre las innovaciones de Gosling ayudaría a largo plazo tanto a Stallman como al movimiento de software libre. Esta confrontación obligaría a Stallman a solucionar las debilidades de la Comuna de Emacs y el sistema informal de confianza que había permitido la aparición se situaciones problemáticas, además, Stallman se vería en la necesidad de endurecer los objetivos políticos del movimiento de software libre. Luego del lanzamiento de GNU Emacs en 1985, Stallman hizo público el "Manifiesto GNU," una extensión del anuncio original, publicado en Septiembre de 1983. Stallman incluyó dentro del documento una larga sección dedicada a los muchos argumentos utilizados por programadores comerciales y académicos para justificar la proliferación de programas de software propietario. Un argumento, "¿Acaso los programadores no merecen una recompensa por su creatividad?, " obtuvo una respuesta que resume la furia de Stallman respecto al reciente episodio con el Emacs de Gosling:

"Si algo merece una recompensa, es la contribución social," escribió Stallman. "La creatividad puede ser una contribución social, pero solo en la medida en que la sociedad sea libre para usar los resultados. Si los programadores merecen ser recompensados por crear programas innovadores, entonces por la misma razón merecen ser castigados si restringen el uso de estos programas." [22]

Con el lanzamiento de GNU Emacs, el Proyecto GNU finalmente tenía código para mostrar. También tenía los problemas típicos de cualquier iniciativa de software. A medida que más y más desarrolladores de Unix comenzaron a jugar con el software, dinero, regalos y solicitudes de cintas comenzaron a llegar. Para atender la parte de negocios del proyecto GNU Stallman reclutó a algunos de sus colegas y formó la Fundación para el Software Libre (*Free Software Foundation*), una organización sin ánimo de lucro dedicada a acelerar el Proyecto GNU hacía su meta. Con Stallman como presidente y varios hackers aliados como miembros del directorio, la FSF ayudó a proporcionarle un rostro corporativo al proyecto GNU.

Robert Chassell, un programador que trabajaba en ese entonces en Lisp Machines, Inc., se convirtió uno de los cinco miembros del directorio en la Fundación para el Software Libre tras una conversación durante una comida con Stallman. Chassell también ejerció como tesorero de la organización, un papel que comenzó siendo pequeño, pero rápidamente creció.

"Creó que en el año 85 nuestros gastos e ingresos totales fueron del orden de $23,000, más o menos," recuerda Chassell. "Richard tenía su oficina, y nosotros tomábamos espacio prestado. Yo puse todas las cosas, especialmente las cintas, bajó mi escritorio. No fue sino hasta algún tiempo después que LMI nos prestó algo de espacio para almecenar cintas y cosas de ese tipo."

Además de proporcionar un rostro, la Fundación para el Software Libre proporcionó un centro de gravedad para otros programadores desencantados. El mercado de Unix, que había parecido tan colegial, incluso en el momento del anuncio inicial de GNU, estaba volviéndose cada vez más competitivo. En un intento por aferrarse de manera mas segura a sus clientes, las compañías habían comenzado a cerrar el acceso al código fuente de Unix. Esta tendencia no hizo más que aumentar el número de solicitudes para los proyectos de software de GNU. Los magos de Unix que alguna vez habían visto a Stallman como un ruidoso agitador de los grupos de noticias, sin demasiados vínculos con la realidad, ahora lo veían como un profeta, una especie de Cassandra del software.

"Mucha gente no se da cuenta, hasta que le toca padecerlo, que tan frustrante pude ser gastar varios años trabajando en un proyecto de software, solo para que te lo quiten después," dice Chassell, resumiendo los sentimientos y opiniones de los corresponsales que escribían a la FSF durante los primeros años. "Después de que eso sucede un par de veces, empiezas a decirte a ti mismo, ‘Oye, espera un momento.’"

Para Chassell, la decisión de participar en la Fundación para el Software Libre se originó en su propio sentimiento personal de pérdida. Antes de trabajar en LMI, Chassell había estado trabajando como empleado, escribiendo un libro introductorio de Unix para Cadmus, Inc. una compañía de Software del área de Cambridge. Cuando Cadmus quebró, arrastrando los derechos del libro consigo, Chassell dice que intentó recomprar los derechos de libro de vuelta, sin ningún éxito.

"Hasta donde yo se, ese libro esta aún apilado en un estante en alguna parte, inutilizable, incopiable, simplemente sustraído del sistema," dice Chassell. "Era una muy buena introducción, si me permiten decirlo. Habría tomado tal vez tres o cuatro meses convertir [el libro] en una introducción a GNU/Linux perfectamente utilizable. La experiencia, con excepción de lo que tengo en mi memoria, se perdió."

Obligado a ver su trabajo hundirse mientras su antiguo empleador pasaba por la bancarrota, Chassell dice que tuvo una idea de la furia que llevó a Stallman al borde de la apoplejía. "La noción más clara, para mi, era el saber que si uno desea llevar una vida decente, entonces no puede tener partes de ella cerradas," dice Chassell. "La idea de tener la libertad de ir y arreglar o modificar una cosa, cualquiera que esta sea, realmente hace la diferencia. Lo hace a uno pensar con alegría que después de haber vivido unos años lo que se ha hecho vale la pena. Porqué de otra manera simplemente se pierde y algún otro se lo lleva y lo deshecha, o lo abandona, y, en el mejor de los casos, uno ya no tiene ninguna relación con eso. Es como perder una parte de la vida."

Notas

[1] Ver Richard Stallman, "Anuncio Inicial de GNU" (Septiembre 1983). http://www.gnu.ai.mit.edu/gnu/initial-announcement.html

[2] Ver Richard Stallman, "Anuncio Inicial de GNU" (Septiembre 1983). http://www.gnu.ai.mit.edu/gnu/initial-announcement.html

[3] Ver Richard Stallman, "Anuncio Inicial de GNU" (Septiembre 1983). http://www.gnu.ai.mit.edu/gnu/initial-announcement.html

[4] Vea Marshall Kirk McKusick, "Twenty Years of Berkeley Unix," [Open Sources] (O'Reilly & Associates, Inc., 1999): 38.

[5] Ver Richard Stallman (1986).

[6] Ver Richard Stallman (1986).

[7] Múltiples fuentes: vea la entrevista con Richard Stallman, el correo de Gerald Sussman, y el Jargon File 3.0.0. http://www.clueless.com/jargon3.0.0/TWENEX.html

[8] Ver http://www.as.cmu.edu/~geek/humor/See_Figure_1.txt

[9] Ver Richard Stallman (1986).

[10] Ver "MIT AI Lab Tourist Policy." http://catalog.com/hopkins/text/tourist-policy.html

[11] Ver Richard Stallman (1986).

[12] Ver Richard Stallman (1986).

[13] Ver H. P. Newquist, [The Brain Makers: Genius, Ego, and Greed in the Quest for Machines that Think] (Sams Publishing, 1994): 172.

[14] [Ibid.]: 196.

[15] [Ibid.] Newquist, quién dice que esta anécdota fue confirmada por varios ejecutivos de Symbolics, escribe, "El mensaje causó una breve ráfaga de excitación y especulación por parte de los empleados de Symbolics, pero al fin y al cabo, nadie tomo el estallido de Stallman tan seriamente."

[16] Ver http://www.as.cmu.edu/~geek/humor/See_Figure_1.txt

[17] Vea Steven Levy, [Hackers] (Penguin USA [paperback], 1984): 426.

[18] Ver Bill Gates, "An Open Letter to Hobbyists" (Febrero 3, 1976). Para ver una copia en línea diríjase a http://www.blinkenlights.com/classiccmp/gateswhine.html.

[19] Ver Richard Stallman, [Open Sources] (O'Reilly & Associates, Inc., 1999): 56. Stallman agrega su propia nota de pie de página, escribiendo, "Como un ateo no sigo ningún líder religioso, pero algunas veces encuentro que admiro algo que alguno de ellos ha dicho."

[20] Ver Richard Stallman (1986).

[21] Vea Richard Stallman (1986).

[22] Ver Richard Stallman, "El Manifesto GNU" (1985). http://www.gnu.org/manifesto.html

Capítulo 8. St. Ignucius

The Maui High Performance Computing Center is located in a single-story building in the dusty red hills just above the town of Kihei. Framed by million-dollar views and the multimillion dollar real estate of the Silversword Golf Course, the center seems like the ultimate scientific boondoggle. Far from the boxy, sterile confines of Tech Square or even the sprawling research metropolises of Argonne, Illinois and Los Alamos, New Mexico, the MHPCC seems like the kind of place where scientists spend more time on their tans than their post-doctoral research projects.

The image is only half true. Although researchers at the MHPCC do take advantage of the local recreational opportunities, they also take their work seriously. According to Top500.org, a web site that tracks the most powerful supercomputers in the world, the IBM SP Power3 supercomputer housed within the MHPCC clocks in at 837 billion floating-point operations per second, making it one of 25 most powerful computers in the world. Co-owned and operated by the University of Hawaii and the U.S. Air Force, the machine divides its computer cycles between the number crunching tasks associated with military logistics and high-temperature physics research.

Simply put, the MHPCC is a unique place, a place where the brainy culture of science and engineering and the laid-back culture of the Hawaiian islands coexist in peaceful equilibrium. A slogan on the lab’s 2000 web site sums it up: "Computing in paradise."

It’s not exactly the kind of place you’d expect to find Richard Stallman, a man who, when taking in the beautiful view of the nearby Maui Channel through the picture windows of a staffer’s office, mutters a terse critique: "Too much sun." Still, as an emissary from one computing paradise to another, Stallman has a message to deliver, even if it means subjecting his pale hacker skin to the hazards of tropical exposure.

The conference room is already full by the time I arrive to catch Stallman’s speech. The gender breakdown is a little better than at the New York speech, 85% male, 15% female, but not by much. About half of the audience members wear khaki pants and logo-encrusted golf shirts. The other half seems to have gone native. Dressed in the gaudy flower-print shirts so popular in this corner of the world, their faces are a deep shade of ochre. The only residual indication of geek status are the gadgets: Nokia cell phones, Palm Pilots, and Sony VAIO laptops.

Needless to say, Stallman, who stands in front of the room dressed in plain blue T-shirt, brown polyester slacks, and white socks, sticks out like a sore thumb. The fluorescent lights of the conference room help bring out the unhealthy color of his sun-starved skin. His beard and hair are enough to trigger beads of sweat on even the coolest Hawaiian neck. Short of having the words "mainlander" tattooed on his forehead, Stallman couldn’t look more alien if he tried.

As Stallman putters around the front of the room, a few audience members wearing T-shirts with the logo of the Maui FreeBSD Users Group (MFUG) race to set up camera and audio equipment. FreeBSD, a free software offshoot of the Berkeley Software Distribution, the venerable 1970s academic version of Unix, is technically a competitor to the GNU/Linux operating system. Still, in the hacking world, Stallman speeches are documented with a fervor reminiscent of the Grateful Dead and its legendary army of amateur archivists. As the local free software heads, it’s up to the MFUG members to make sure fellow programmers in Hamburg, Mumbai, and Novosibirsk don’t miss out on

the latest pearls of RMS wisdom.

The analogy to the Grateful Dead is apt. Often, when describing the business opportunities inherent within the free software model, Stallman has held up the Grateful Dead as an example. In refusing to restrict fans' ability to record live concerts, the Grateful Dead became more than a rock group. They became the center of a tribal community dedicated to Grateful Dead music. Over time, that tribal community became so large and so devoted that the band shunned record contracts and supported itself solely through musical tours and live appearances. In 1994, the band's last year as a touring act, the Grateful Dead drew $52 million in gate receipts alone. [1]

While the tribal aspect of the free software community is one reason many in the latter half of the 1990s started to accept the notion that publishing software source code might be a good thing. Hoping to build their own loyal followings, companies such as IBM, Sun Microsystems, and Hewlett Packard have come to accept the letter, if not the spirit, of the Stallman free software message. Describing the GPL as the information-technology industry's "Magna Carta," ZDNet software columnist Evan Leibovitch sees the growing affection for all things GNU as more than just a trend. "This societal shift is letting users take back control of their futures," Leibovitch writes. "Just as the Magna Carta gave rights to British subjects, the GPL enforces consumer rights and freedoms on behalf of the users of computer software." [2]

The tribal aspect of the free software community also helps explain why 40-odd programmers, who might otherwise be working on physics projects or surfing the Web for windsurfing buoy reports, have packed into a conference room to hear Stallman speak.

Unlike the New York speech, Stallman gets no introduction. He also offers no self-introduction. When the FreeBSD people finally get their equipment up and running, Stallman simply steps forward, starts speaking, and steamrolls over every other voice in the room.

"Most of the time when people consider the question of what rules society should have for using software, the people considering it are from software companies, and they consider the question from a self-serving perspective," says Stallman, opening his speech. "What rules can we impose on everybody else so they have to pay us lots of money? I had the good fortune in the 1970s to be part of a community of programmers who shared software. And because of this I always like to look at the same issue from a different direction to ask: what kind of rules make possible a good society that is good for the people who are in it? And therefore I reach completely different answers."

Once again, Stallman quickly segues into the parable of the Xerox laser printer, taking a moment to deliver the same dramatic finger-pointing gestures to the crowd. He also devotes a minute or two to the GNU/Linux name.

"Some people say to me, 'Why make such a fuss about getting credit for this system? After all, the important thing is the job is done, not whether you get recognition for it.' Well, this would be wise advice if it were true. But the job wasn't to build an operating system; the job is to spread freedom to the users of computers. And to do that we have to make it possible to do everything with computers in freedom." [3]

Adds Stallman, "There's a lot more work to do."

For some in the audience, this is old material. For others, it's a little arcane. When a member of the golf-shirt contingent starts dozing off, Stallman stops the speech and asks somebody to wake the person up.

"Somebody once said my voice was so soothing, he asked if I was some kind of healer," says Stallman, drawing a quick laugh from the crowd. "I guess that probably means I can help you drift gently into a blissful, relaxing sleep. And some of you might need that. I guess I shouldn't object if you do. If you need to sleep, by all means do."

The speech ends with a brief discussion of software patents, a growing issue of concern both within the software industry and within the free software community. Like Napster, software patents reflect the awkward nature of applying laws and concepts written for the physical world to the frictionless universe of information technology. The difference between protecting a program under copyright and protecting a program under software patents is subtle but significant. In the case of copyright, a software creator can restrict duplication of the source code but not duplication of the idea or functionality that the source code addresses. In other words, if a developer chooses not to use a software program under the original developer's terms, that second developer is still free to reverse-engineer the program--i.e., duplicate the software program's functionality by rewriting the source code from scratch. Such duplication of ideas is common within the commercial software industry, where companies often isolate reverse-engineering teams to head off accusations of corporate espionage or developer hanky-panky. In the jargon of modern software development, companies refer to this technique as "clean room" engineering.

Software patents work differently. According to the U.S. Patent Office, companies and individuals may secure patents for innovative algorithms provided they submit their claims to a public review. In theory, this allows the patent-holder to trade off disclosure of their invention for a limited monopoly of a minimum of 20 years after the patent filing. In practice, the disclosure is of limited value, since the operation of the program is often self-evident. Unlike copyright, a patent gives its holder the ability to head off the independent development of software programs with the same or similar functionality.

In the software industry, where 20 years can cover the entire life cycle of a marketplace, patents take on a strategic weight. Where companies such as Microsoft and Apple once battled over copyright and the "look and feel" of various technologies, today's Internet companies use patents as a way to stake out individual applications and business models, the most notorious example being Amazon.com's 2000 attempt to patent the company's "one-click" online shopping process. For most companies, however, software patents have become a defensive tool, with cross-licensing deals balancing one set of corporate patents against another in a tense form of corporate detente. Still, in a few notable cases of computer encryption and graphic imaging algorithms, software vendors have successfully stifled rival technologies.

For Stallman, the software-patent issue dramatizes the need for eternal hacker vigilance. It also underlines the importance of stressing the political benefits of free software programs over the competitive benefits. Pointing to software patents' ability to create sheltered regions in the marketplace, Stallman says competitive performance and price, two areas where free software operating systems such as GNU/Linux and FreeBSD already hold a distinct advantage over their proprietary counterparts, are red herrings compared to the large issues of user and developer freedom.

"It's not because we don't have the talent to make better software," says Stallman. "It's because we don't have the right. Somebody has prohibited us from serving the public. So what's going to happen when users encounter these gaps in free software? Well, if they have been persuaded by the open source movement that these freedoms are good because they lead to more-powerful reliable software, they're likely to say, 'You didn't deliver what you promised. This software's not more powerful. It's missing this feature. You lied to me.' But if they have come to agree with the free software movement, that the freedom is important in itself, then they will say, 'How dare those people stop me from having this feature and my freedom too.' And with that kind of response, we may survive the hits that we're

going to take as these patents explode."

Such comments involve a hefty dose of spin, of course. Most open source advocates are equally, if not more, vociferous as Stallman when it comes to opposing software patents. Still, the underlying logic of Stallman's argument--that open source advocates emphasize the utilitarian advantages of free software over the political advantages--remains uncontested. Rather than stress the political significance of free software programs, open source advocates have chosen to stress the engineering integrity of the hacker development model. Citing the power of peer review, the open source argument paints programs such as GNU/Linux or FreeBSD as better built, better inspected and, by extension, more trushworthy to the average user.

That's not to say the term "open source" doesn't have its political implications. For open source advocates, the term open source serves two purposes. First, it eliminates the confusion associated with the word "free," a word many businesses interpret as meaning "zero cost." Second, it allows companies to examine the free software phenomenon on a technological, rather than ethical, basis. Eric Raymond, cofounder of the Open Source Initiative and one of the leading hackers to endorse the term, effectively summed up the frustration of following Stallman down the political path in a 1999 essay, titled "Shut Up and Show Them the Code":

> RMS's rhetoric is very seductive to the kind of people we are. We hackers are thinkers and idealists who readily resonate with appeals to "principle" and "freedom" and "rights." Even when we disagree with bits of his program, we want RMS's rhetorical style to work; we think it ought to work; we tend to be puzzled and disbelieving when it fails on the 95% of people who aren't wired like we are. [4]

Included among that 95%, Raymond writes, are the bulk of business managers, investors, and nonhacker computer users who, through sheer weight of numbers, tend to decide the overall direction of the commercial software marketplace. Without a way to win these people over, Raymond argues, programmers are doomed to pursue their ideology on the periphery of society:

> When RMS insists that we talk about "computer users' rights," he's issuing a dangerously attractive invitation to us to repeat old failures. It's one we should reject--not because his principles are wrong, but because that kind of language, applied to software, simply does not persuade anybody but us. In fact, it confuses and repels most people outside our culture. [5]

Watching Stallman deliver his political message in person, it is hard to see anything confusing or repellent. Stallman's appearance may seem off-putting, but his message is logical. When an audience member asks if, in shunning proprietary software, free software proponents lose the ability to keep up with the latest technological advancements, Stallman answers the question in terms of his own personal beliefs. "I think that freedom is more important than mere technical advance," he says. "I would always choose a less advanced free program rather than a more advanced nonfree program, because I won't give up my freedom for something like that. My rule is, if I can't share it with you, I won't take it."

Such answers, however, reinforce the quasi-religious nature of the Stallman message. Like a Jew keeping kosher or a Mormon refusing to drink alcohol, Stallman paints his decision to use free software in the place of proprietary in the color of tradition and personal belief. As software evangelists go, Stallman avoids forcing those beliefs down listeners' throats. Then again, a listener rarely leaves a Stallman speech not knowing where the true path to software righteousness lies.

As if to drive home this message, Stallman punctuates his speech with an unusual ritual. Pulling a black robe out of a plastic grocery bag, Stallman puts it on. Out of a second bag, he pulls a reflective yellow computer disk and places it on his head. The crowd lets out a startled laugh.

"I am St. Ignucius of the Church of Emacs," says Stallman, raising his right hand in mock-blessing. "I bless your computer, my child."

The laughter turns into full-blown applause after a few seconds. As audience members clap, the computer disk on Stallman's head catches the glare of an overhead light, eliciting a perfect halo effect. In the blink of an eye, Stallman goes from awkward *haole* to Russian religious icon.

"Emacs was initially a text editor," says Stallman, explaining the getup. "Eventually it became a way of life for many and a religion for some. We call this religion the Church of Emacs."

The skit is a lighthearted moment of self-pardoy, a humorous return-jab at the many people who might see Stallman's form of software asceticism as religious fanaticism in disguise. It is also the sound of the other shoe dropping-loudly. It's as if, in donning his robe and halo, Stallman is finally letting listeners of the hook, saying, "It's OK to laugh. I know I'm weird."

Discussing the St. Ignucius persona afterward, Stallman says he first came up with it in 1996, long after the creation of Emacs but well before the emergence of the "open source" term and the struggle for hacker-community leadership that precipitated it. At the time, Stallman says, he wanted a way to "poke fun at himself," to remind listeners that, though stubborn, Stallman was not the fanatic some made him out to be. It was only later, Stallman adds, that others seized the persona as a convenient way to play up his reputation as software ideologue, as Eric Raymond did in an 1999 interview with the linux.com web site:

> When I say RMS calibrates what he does, I'm not belittling or accusing him of insincerity. I'm saying that like all good communicators he's got a theatrical streak. Sometimes it's conscious-have you ever seen him in his St. Ignucius drag, blessing software with a disk platter on his head? Mostly it's unconscious; he's just learned the degree of irritating stimulus that works, that holds attention without (usually) freaking people out. [6]

Stallman takes issue with the Raymond analysis. "It's simply my way of making fun of myself," he says. "The fact that others see it as anything more than that is a reflection of their agenda, not mine."

That said, Stallman does admit to being a ham. "Are you kidding?" he says at one point. "I love being the center of attention." To facilitate that process, Stallman says he once enrolled in Toastmasters, an organization that helps members bolster their public-speaking skills and one Stallman recommends highly to others. He possesses a stage presence that would be the envy of most theatrical performers and feels a link to vaudevillians of years past. A few days after the Maui High Performance Computing Center speech, I allude to the 1999 LinuxWorld performace and ask Stallman if he has a Groucho Marx complex--i.e., the unwillingness to belong to any club that would have him as a member. Stallman's response is immediate: "No, but I admire Groucho Marx in a lot of ways and certainly have been in some things I say inspired by him. But then I've also been inspired in some ways by Harpo."

The Groucho Marx influence is certainly evident in Stallman's lifelong fondness for punning. Then again, punning and wordplay are common hacker traits. Perhaps the most Groucho-like aspect of Stallman's personality, however, is the deadpan manner in which the puns are delivered. Most come so stealthily--without even the hint of a raised eyebrow or upturned smile--you almost have to wonder if Stallman's laughing at his audience more than the audience is laughing at him.

Watching members of the Maui High Performance Computer Center laugh at the St. Ignucius parody, such concerns evaporate. While not exactly a standup act, Stallman certainly possesses the chops to keep a roomful of engineers in stitches. "To be a saint in the Church of Emacs does not require celibacy, but it does require making a commitment to living a life of moral purity," he tells the Maui audience. "You must exorcise the evil proprietary operating system from all your computer and then install a wholly [holy] free operating system. And then you must install only free software on top of that. If you make this commitment and live by it, then you too will be a saint in the Church of Emacs, and you too may have a halo."

The St. Ignucius skit ends with a brief inside joke. On most Unix systems and Unix-related offshoots, the primary competitor program to Emacs is vi, a text-editing program developed by former UC Berkeley student and current Sun Microsystems chief scientist, Bill Joy. Before doffing his "halo," Stallman pokes fun at the rival program. "People sometimes ask me if it is a sin in the Church of Emacs to use vi," he says. "Using a free version of vi is not a sin; it is a penance. So happy hacking."

After a brief question-and-answer session, audience members gather around Stallman. A few ask for autographs. "I'll sign this," says Stallman, holding up one woman's print out of the GNU General Public License, "but only if you promise me to use the term GNU/Linux instead of Linux and tell all your friends to do likewise."

The comment merely confirms a private observation. Unlike other stage performers and political figures, Stallman has no "off" mode. Aside from the St. Ignucius character, the ideologue you see onstage is the ideologue you meet backstage. Later that evening, during a dinner conversation in which a programmer mentions his affinity for "open source" programs, Stallman, between bites, upbraids his tablemate: "You mean free software. That's the proper way to refer to it."

During the question-and-answer session, Stallman admits to playing the pedagogue at times. "There are many people who say, 'Well, first let's invite people to join the community, and then let's teach them about freedom.' And that could be a reasonable strategy, but what we have is almost everybody's inviting people to join the community, and hardly anybody's teaching them about freedom once they come in."

The result, Stallman says, is something akin to a third-world city. People move in, hoping to strike it rich or at the very least to take part in a vibrant, open culture, and yet those who hold the true power keep evolving new tricks and strategies--i.e., software patents--to keep the masses out. "You have millions of people moving in and building shantytowns, but nobody's working on step two: getting them out of those shantytowns. If you think talking about software freedom is a good strategy, please join in doing step two. There are plenty working on step one. We need more people working on step two."

Working on "step two" means driving home the issue that freedom, not acceptance, is the root issue of the free software movement. Those who hope to reform the proprietary software industry from the inside are on a fool's errand. "Change from the inside is risky," Stallman stays. "Unless you're working at the level of a Gorbachev, you're going to be neutralized."

Hands pop up. Stallman points to a member of the golf shirt-wearing contingent. "Without patents, how would you suggest dealing with commercial espionage?"

"Well, those two questions have nothing to do with each other, really," says Stallman.

"But I mean if someone wants to steal another company’s piece of software."

Stallman’s recoils as if hit by a poisonous spray. "Wait a second," Stallman says. "Steal? I’m sorry, there’s so much prejudice in that statement that the only thing I can say is that I reject that prejudice. Companies that develop nonfree software and other things keep lots and lots of trade secrets, and so that’s not really likely to change. In the old days--even in the 1980s--for the most part programmers were not aware that there were even software patents and were paying no attention to them. What happened was that people published the interesting ideas, and if they were not in the free software movement, they kept secret the little details. And now they patent those broad ideas and keep secret the little details. So as far as what you’re describing, patents really make no difference to it one way or another."

"But if it doesn’t affect their publication," a new audience member jumps in, his voice trailing off almost as soon as he starts speaking.

"But it does," Stallman says. "Their publication is telling you that this is an idea that’s off limits to the rest of the community for 20 years. And what the hell good is that? Besides, they’ve written it in such a hard way to read, both to obfuscate the idea and to make the patent as broad as possible, that it’s basically useless looking at the published information to learn anything anyway. The only reason to look at patents is to see the bad news of what you can’t do."

The audience falls silent. The speech, which began at 3:15, is now nearing the 5:00 whistle, and most listeners are already squirming in their seats, antsy to get a jump start on the weekend. Sensing the fatigue, Stallman glances around the room and hastily shuts things down. "So it looks like we’re done," he says, following the observation with an auctioneer’s "going, going, gone" to flush out any last-minute questioners. When nobody throws their hand up, Stallman signs off with a traditional exit line.

"Happy hacking," he says.

Notas

[1] See "Grateful Dead Time Capsule: 1985-1995 North American Tour Grosses." http://www.accessplace.com/gdtc/1197.htm

[2] See Evan Leibovitch, "Who's Afraid of Big Bad Wolves," [ZDNet Tech Update] (December 15, 2000). http://techupdate.zdnet.com/techupdate/stories/main/014179266499200.html

[3] For narrative purposes, I have hesitated to go in-depth when describing Stallman's full definition of software "freedom." The GNU Project web site lists four fundamental components:

- The freedom to run a program, for any purpose (freedom 0).
- The freedom to study how a program works, and adapt it to your needs (freedom 1).
- The freedom to redistribute copies of a program so you can help your neighbor (freedom 2).
- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3).

For more information, please visit "The Free Software Definition" athttp://www.gnu.org/philosophy/free-sw.html.

[4] See Eric Raymond, "Shut Up and Show Them the Code," online essay, (June 28, 1999).

[5] See Eric Raymond, "Shut Up and Show Them the Code," online essay, (June 28, 1999).

[6] See "Guest Interview: Eric S. Raymond," [Linux.com] (May 18, 1999). http://www.linux.com/interviews/19990518/8/

Capítulo 9. La Licencia Pública General GNU

En la primavera de 1985, Richard Stallman se había concentrado en el primer gran logro del movimiento GNU: una versión de Emacs libre basada en Lisp. Para cumplir este objetivo, sin embargo, debía enfrentarse a dos grandes retos. En primer lugar, tenía que encontrar la manera de reconstruir Emacs de forma tal que no fuera dependiente de ninguna plataforma. En segundo, tenía que reconstruir la Comuna de Emacs de manera análoga.

La disputa con UniPress había revelado una debilidad en el contrato social de la Cumuna de Emacs. Aúnque los usuarios obtenían la experiencia y visión de Stallman, las reglas de la Comuna se impusieron. En áreas en las que Stallman no mantenía su posición de alpha hacker -en sistemas Unix anteriores a 1984, por ejemplo-, otras personas y compañías tenían la libertad de crear sus propias reglas.

La tensión entre la libertad de modificar y la libertad de ejercer privilegios autoritarios había estado creciendo en el caso de GOSMACS. El Acto de Derechos de Autor de 1976 había extendido las leyes de derechos de autor en los Estados Unidos, aumentando su cobertura para incluir los programas de computador. De acuerdo con la sección 102(b) del Acto, se les daba a las personas y compañías la capacidad de proteger la "expresión" de un programa pero no los "procesos o métodos utilizados en el programa". [1] En la práctica, esto les permitía a los programadores y a las compañías tratar los programas de computadores como novelas o canciones. Otros programadores podrían inspirarse en las obras, pero para realizar una copia directa o una obra derivada (que no fuera de naturaleza satírica) tendrían que obtener un permiso explícito del autor. Aunque la nueva legislación garantizaba que aún programas sin advertencias de reservas sobre los derechos de autor se veían protegidos, los programadores no tardaron en asegurar sus derechos, adjuntando advertencias al respecto a sus programas.

Al principio, Stallman reaccionó alarmado a estas advertencia. En esa época eran poco comunes los programas que no compartieran código fuente con otros. bastó una simple firma del presidente para que el congreso les diera a los programadores y compañías el poder de asegurar su propiedad individual sobre programas creados por comunidades. Además se inyectó una dosis de formalidad en un sistema que había sido siempre muy informal. Aún si los hackers pudiesen haber demostrado que el código fuente de un programa determinado provenía de años o décadas de trabajo, no tenían los recursos y el dinero que hubieran requerido para combatir estas advertencias de derechos de autor. Resumiendo, estas disputas que antes habían sido arregladas entre hackers ahora tendrían que ser resueltas por abogados. Así las cosas, las compañías, no los hackers, tenían una gran ventaja.

Los promotores de los derechos de autor en el software tenían sus propios argumentos: sin derechos de autor, las obras pasarían directamente al dominio público. Colocar una advertencia de derechos de autor en una obra también servía como una muestra de calidad. Los programadores o compañías que colocaban sus nombres en estas advertencias colocaban también su reputación. Finalmente, Proponents of software copyright had their counter-arguments: without copyright, works might otherwise slip into the public domain. Putting a copyright notice on a work also served as a statement of quality. Programmers or companies who attached their name to the copyright attached their reputations as well. Finally, it was a contract, as well as a statement of ownership. Using copyright as a flexible form of license, an author could give away certain rights in exchange for certain forms of

behavior on the part of the user. For example, an author could give away the right to suppress unauthorized copies just so long as the end user agreed not to create a commercial offshoot.

It was this last argument that eventually softened Stallman's resistance to software copyright notices. Looking back on the years leading up to the GNU Project, Stallman says he began to sense the beneficial nature of copyright sometime around the release of Emacs 15.0, the last significant pre-GNU Project upgrade of Emacs. "I had seen email messages with copyright notices plus simple 'verbatim copying permitted' licenses," Stallman recalls. "Those definitely were [an] inspiration."

For Emacs 15, Stallman drafted a copyright that gave users the right to make and distribute copies. It also gave users the right to make modified versions, but not the right to claim sole ownership of those modified versions, as in the case of GOSMACS.

Although helpful in codifying the social contract of the Emacs Commune, the Emacs 15 license remained too "informal" for the purposes of the GNU Project, Stallman says. Soon after starting work on a GNU version of Emacs, Stallman began consulting with the other members of the Free Software Foundation on how to shore up the license's language. He also consulted with the attorneys who had helped him set up the Free Software Foundation.

Mark Fischer, a Boston attorney specializing in intellectual-property law, recalls discussing the license with Stallman during this period. "Richard had very strong views about how it should work," Fischer says, "He had two principles. The first was to make the software absolutely as open as possible. The second was to encourage others to adopt the same licensing practices."

Encouraging others to adopt the same licensing practices meant closing off the escape hatch that had allowed privately owned versions of Emacs to emerge. To close that escape hatch, Stallman and his free software colleagues came up with a solution: users would be free to modify GNU Emacs just so long as they published their modifications. In addition, the resulting "derivative" works would also have carry the same GNU Emacs License.

The revolutionary nature of this final condition would take a while to sink in. At the time, Fischer says, he simply viewed the GNU Emacs License as a simple contract. It put a price tag on GNU Emacs' use. Instead of money, Stallman was charging users access to their own later modifications. That said, Fischer does remember the contract terms as unique.

"I think asking other people to accept the price was, if not unique, highly unusual at that time," he says.

The GNU Emacs License made its debut when Stallman finally released GNU Emacs in 1985. Following the release, Stallman welcomed input from the general hacker community on how to improve the license's language. One hacker to take up the offer was future software activist John Gilmore, then working as a consultant to Sun Microsystems. As part of his consulting work, Gilmore had ported Emacs over to SunOS, the company's in-house version of Unix. In the process of doing so, Gilmore had published the changes as per the demands of the GNU Emacs License. Instead of viewing the license as a liability, Gilmore saw it as clear and concise expression of the hacker ethos. "Up until then, most licenses were very informal," Gilmore recalls.

As an example of this informality, Gilmore cites a copyright notice for trn, a Unix utility. Written by Larry Wall, future creator of the Perl programming language, patch made it simple for Unix programmers to insert source-code fixes--"patches" in hacker jargon--into any large program. Recognizing the utility of this feature, Wall put the following copyright notice in the program's accompanying README file:

```
Copyright (c) 1985, Larry Wall You may copy the trn
    kit in whole or in part as long as you don’t try to make money off
    it, or pretend that you wrote it.
[2]
```

Such statements, while reflective of the hacker ethic, also reflected the difficulty of translating the loose, informal nature of that ethic into the rigid, legal language of copyright. In writing the GNU Emacs License, Stallman had done more than close up the escape hatch that permitted proprietary offshoots. He had expressed the hacker ethic in a manner understandable to both lawyer and hacker alike.

It wasn’t long, Gilmore says, before other hackers began discussing ways to "port" the GNU Emacs License over to their own programs. Prompted by a conversation on Usenet, Gilmore sent an email to Stallman in November, 1986, suggesting modification:

> You should probably remove "EMACS" from the license and replace it with "SOFTWARE" or something. Soon, we hope, Emacs will not be the biggest part of the GNU system, and the license applies to all of it. [3]

Gilmore wasn’t the only person suggesting a more general approach. By the end of 1986, Stallman himself was at work with GNU Project’s next major milestone, a source-code debugger, and was looking for ways to revamp the Emacs license so that it might apply to both programs. Stallman’s solution: remove all specific references to Emacs and convert the license into a generic copyright umbrella for GNU Project software. The GNU General Public License, GPL for short, was born.

In fashioning the GPL, Stallman followed the software convention of using decimal numbers to indicate prototype versions and whole numbers to indicate mature versions. Stallman published Version 1.0 of the GPL in 1989 (a project Stallman was developing in 1985), almost a full year after the release of the GNU Debugger, Stallman’s second major foray into the realm of Unix programming. The license contained a preamble spelling out its political intentions:

```
The General Public License is designed to make sure that you have
the freedom to give away or sell copies of free software, that you
receive source code or can get it if you want it, that you can change
the software or use pieces of it in new free programs; and that you
know you can do these things.

To protect your rights, we need to make restrictions that forbid
anyone to deny you these rights or to ask you to surrender the
rights. These restrictions translate to certain responsibilities for
you if you distribute copies of the software, or if you modify
it.
[4]
```

In fashioning the GPL, Stallman had been forced to make an additional adjustment to the informal tenets of the old Emacs Commune. Where he had once demanded that Commune members publish any and all changes, Stallman now demanded publication only in instances when programmers circulated their derivative versions in the same public manner as Stallman. In other words, programmers who simply modified Emacs for private use no longer needed to send the source-code changes back to Stallman. In what would become a rare compromise of free software doctrine, Stallman slashed the price tag for free software. Users could innovate without Stallman looking over their shoulders just so long as they didn’t bar Stallman and the rest of the hacker community from future exchanges of the same program.

Looking back, Stallman says the GPL compromise was fueled by his own dissatisfaction with the Big Brother aspect of the original Emacs Commune social contract. As much as he liked peering into other hackers' systems, the knowledge that some future source-code maintainer might use that power to ill effect forced him to temper the GPL.

"It was wrong to require people to publish all changes," says Stallman. "It was wrong to require them to be sent to one privileged developer. That kind of centralization and privilege for one was not consistent with a society in which all had equal rights."

As hacks go, the GPL stands as one of Stallman's best. It created a system of communal ownership within the normally proprietary confines of copyright law. More importantly, it demonstrated the intellectual similarity between legal code and software code. Implicit within the GPL's preamble was a profound message: instead of viewing copyright law with suspicion, hackers should view it as yet another system begging to be hacked.

"The GPL developed much like any piece of free software with a large community discussing its structure, its respect or the opposite in their observation, needs for tweaking and even to compromise it mildly for greater acceptance," says Jerry Cohen, another attorney who helped Stallman with the creation of the license. "The process worked very well and GPL in its several versions has gone from widespread skeptical and at times hostile response to widespread acceptance."

In a 1986 interview with [Byte] magazine, Stallman summed up the GPL in colorful terms. In addition to proclaiming hacker values, Stallman said, readers should also "see it as a form of intellectual jujitsu, using the legal system that software hoarders have set up against them." [5] Years later, Stallman would describe the GPL's creation in less hostile terms. "I was thinking about issues that were in a sense ethical and in a sense political and in a sense legal," he says. "I had to try to do what could be sustained by the legal system that we're in. In spirit the job was that of legislating the basis for a new society, but since I wasn't a government, I couldn't actually change any laws. I had to try to do this by building on top of the existing legal system, which had not been designed for anything like this."

About the time Stallman was pondering the ethical, political, and legal issues associated with free software, a California hacker named Don Hopkins mailed him a manual for the 68000 microprocessor. Hopkins, a Unix hacker and fellow science-fiction buff, had borrowed the manual from Stallman a while earlier. As a display of gratitude, Hopkins decorated the return envelope with a number of stickers obtained at a local science-fiction convention. One sticker in particular caught Stallman's eye. It read, "Copyleft (L), All Rights Reversed." Following the release of the first version of GPL, Stallman paid tribute to the sticker, nicknaming the free software license "Copyleft." Over time, the nickname and its shorthand symbol, a backwards "C," would become an official Free Software Foundation synonym for the GPL.

The German sociologist Max Weber once proposed that all great religions are built upon the "routinization" or "institutionalization" of charisma. Every successful religion, Weber argued, converts the charisma or message of the original religious leader into a social, political, and ethical apparatus more easily translatable across cultures and time.

While not religious per se, the GNU GPL certainly qualifies as an interesting example of this "routinization" process at work in the modern, decentralized world of software development. Since its unveiling, programmers and companies who have otherwise expressed little loyalty or allegiance to Stallman have willingly accepted the GPL bargain at face value. A few have even accepted the GPL as a preemptive protective mechanism for their own software programs. Even those who reject the GPL contract as too compulsory, still credit it as influential.

One hacker falling into this latter group was Keith Bostic, a University of California employee at the time of the GPL 1.0 release. Bostic's department, the Computer Systems Research Group (SRG), had been involved in Unix development since the late 1970s and was responsible for many key parts of Unix, including the TCP/IP networking protocol, the cornerstone of modern Internet communications. By the late 1980s, AT&T, the original owner of the Unix brand name, began to focus on commercializing Unix and began looking to the Berkeley Software Distribution, or BSD, the academic version of Unix developed by Bostic and his Berkeley peers, as a key source of commercial technology.

Although the Berkeley BSD source code was shared among researchers and commercial programmers with a source-code license, this commercialization presented a problem. The Berkeley code was intermixed with proprietary AT&T code. As a result, Berkeley distributions were available only to institutions that already had a Unix source license from AT&T. As AT&T raised its license fees, this arrangement, which had at first seemed innocuous, became increasingly burdensome.

Hired in 1986, Bostic had taken on the personal project of porting BSD over to the Digital Equipment Corporation's PDP-11 computer. It was during this period, Bostic says, that he came into close interaction with Stallman during Stallman's occasional forays out to the west coast. "I remember vividly arguing copyright with Stallman while he sat at borrowed workstations at CSRG," says Bostic. "We'd go to dinner afterward and continue arguing about copyright over dinner."

The arguments eventually took hold, although not in the way Stallman would have liked. In June, 1989, Berkeley separated its networking code from the rest of the AT&T-owned operating system and distributed it under a University of California license. The contract terms were liberal. All a licensee had to do was give credit to the university in advertisements touting derivative programs. [6] In contrast to the GPL, proprietary offshoots were permissible. Only one problem hampered the license's rapid adoption: the BSD Networking release wasn't a complete operating system. People could study the code, but it could only be run in conjunction with other proprietary-licensed code.

Over the next few years, Bostic and other University of California employees worked to replace the missing components and turn BSD into a complete, freely redistributable operating system. Although delayed by a legal challenge from Unix Systems Laboratories--the AT&T spin-off that retained ownership of the Unix brand name--the effort would finally bear fruit in the early 1990s. Even before then, however, many of the Berkeley utilities would make their way into Stallman's GNU Project.

"I think it's highly unlikely that we ever would have gone as strongly as we did without the GNU influence," says Bostic, looking back. "It was clearly something where they were pushing hard and we liked the idea."

By the end of the 1980s, the GPL was beginning to exert a gravitational effect on the free software community. A program didn't have to carry the GPL to qualify as free software-witness the case of the BSD utilities--but putting a program under the GPL sent a definite message. "I think the very existence of the GPL inspired people to think through whether they were making free software, and how they would license it," says Bruce Perens, creator of Electric Fence, a popular Unix utility, and future leader of the Debian GNU/Linux development team. A few years after the release of the GPL, Perens says he decided to discard Electric Fence's homegrown license in favor of Stallman's lawyer-vetted copyright. "It was actually pretty easy to do," Perens recalls.

Rich Morin, the programmer who had viewed Stallman's initial GNU announcement with a degree of skepticism, recalls being impressed by the software that began to gather under the GPL umbrella. As the leader of a SunOS user group, one of Morin's primary duties during the 1980s had been to send out distribution tapes containing the best freeware or free software utilities. The job often mandated

calling up original program authors to verify whether their programs were copyright protected or whether they had been consigned to the public domain. Around 1989, Morin says, he began to notice that the best software programs typically fell under the GPL license. "As a software distributor, as soon as I saw the word GPL, I knew I was home free," recalls Morin.

To compensate for the prior hassles that went into compiling distribution tapes to the Sun User Group, Morin had charged recipients a convenience fee. Now, with programs moving over to the GPL, Morin was suddenly getting his tapes put together in half the time, turning a tidy profit in the process. Sensing a commercial opportunity, Morin rechristened his hobby as a business: Prime Time Freeware.

Such commercial exploitation was completely within the confines of the free software agenda. "When we speak of free software, we are referring to freedom, not price," advised Stallman in the GPL's preamble. By the late 1980s, Stallman had refined it to a more simple mnemonic: "Don't think free as in free beer; think free as in free speech."

For the most part, businesses ignored Stallman's entreaties. Still, for a few entrepreneurs, the freedom associated with free software was the same freedom associated with free markets. Take software ownership out of the commercial equation, and you had a situation where even the smallest software company was free to compete against the IBMs and DECs of the world.

One of the first entrepreneurs to grasp this concept was Michael Tiemann, a software programmer and graduate student at Stanford University. During the 1980s, Tiemann had followed the GNU Project like an aspiring jazz musician following a favorite artist. It wasn't until the release of the GNU C Compiler in 1987, however, that he began to grasp the full potential of free software. Dubbing GCC a "bombshell," Tiemann says the program's own existence underlined Stallman's determination as a programmer.

"Just as every writer dreams of writing the great American novel, every programmer back in the 1980s talked about writing the great American compiler," Tiemman recalls. "Suddenly Stallman had done it. It was very humbling."

"You talk about single points of failure, GCC was it," echoes Bostic. "Nobody had a compiler back then, until GCC came along."

Rather than compete with Stallman, Tiemann decided to build on top of his work. The original version of GCC weighed in at 110,000 lines of code, but Tiemann recalls the program as surprisingly easy to understand. So easy in fact that Tiemann says it took less than five days to master and another week to port the software to a new hardware platform, National Semiconductor's 32032 microchip. Over the next year, Tiemann began playing around with the source code, creating a native compiler for the C+ programming language. One day, while delivering a lecture on the program at Bell Labs, Tiemann ran into some AT&T developers struggling to pull off the same thing.

"There were about 40 or 50 people in the room, and I asked how many people were working on the native code compiler," Tiemann recalls. "My host said the information was confidential but added that if I took a look around the room I might get a good general idea."

It wasn't long after, Tiemann says, that the light bulb went off in his head. "I had been working on that project for six months," Tiemann says. "I just thought to myself, whether it's me or the code this is a level of efficiency that the free market should be ready to reward."

Tiemann found added inspiration in the GNU Manifesto, which, while excoriating the greed of some software vendors, encourages other vendors to consider the advantages of free software from a consumer point of view. By removing the power of monopoly from the commerical software question, the GPL makes it possible for the smartest vendors to compete on the basis of service and consulting, the two most profit-rich corners of the software marketplace.

In a 1999 essay, Tiemann recalls the impact of Stallman's Manifesto. "It read like a socialist polemic, but I saw something different. I saw a business plan in disguise." [7]

Teaming up with John Gilmore, another GNU Project fan, Tiemann launched a software consulting service dedicated to customizing GNU programs. Dubbed Cygnus Support, the company signed its first development contract in February, 1990. By the end of the year, the company had $725,000 worth of support and development contracts.

GNU Emacs, GDB, and GCC were the "big three" of developer-oriented tools, but they weren't the only ones developed by Stallman during the GNU Project's first half decade. By 1990, Stallman had also generated GNU versions of the Bourne Shell (rechristened the Bourne Again Shell, or BASH), YACC (rechristened Bison), and awk (rechristened gawk). Like GCC, every GNU program had to be designed to run on multiple systems, not just a single vendor's platform. In the process of making programs more flexible, Stallman and his collaborators often made them more useful as well.

Recalling the GNU universalist approach, Prime Time Freeware's Morin points to a critical, albeit mundane, software package called hello. "It's the hello world program which is five lines of C, packaged up as if it were a GNU distribution," Morin says. "And so it's got the Texinfo stuff and the configure stuff. It's got all the other software engineering goo that the GNU Project has come up with to allow packages to port to all these different environments smoothly. That's tremendously important work, and it affects not only all of [Stallman's] software, but also all of the other GNU Project software."

According to Stallman, improving software programs was secondary to building them in the first place. "With each piece I may or may not find a way to improve it," said Stallman to [Byte]. "To some extent I am getting the benefit of reimplementation, which makes many systems much better. To some extent it's because I have been in the field a long time and worked on many other systems. I therefore have many ideas to bring to bear." [8]

Nevertheless, as GNU tools made their mark in the late 1980s, Stallman's AI Lab-honed reputation for design fastidiousness soon became legendary throughout the entire software-development community.

Jeremy Allison, a Sun user during the late 1980s and programmer destined to run his own free software project, Samba, in the 1990s, recalls that reputation with a laugh. During the late 1980s, Allison began using Emacs. Inspired by the program's community-development model, Allison says he sent in a snippet of source code only to have it rejected by Stallman.

"It was like the [Onion] headline," Allison says. "'Child's prayers to God answered: No.'"

Stallman's growing stature as a software programmer, however, was balanced by his struggles as a project manager. Although the GNU Project moved from success to success in creation of developer-oriented tools, its inability to generate a working kernel--the central "traffic cop" program in all Unix systems that determines which devices and applications get access to the microprocessor and when--was starting to elicit grumbles as the 1980s came to a close. As with most GNU Project efforts, Stallman had started kernel development by looking for an existing program to modify. According to a January 1987 "Gnusletter," Stallman was already working to overhaul TRIX, a Unix kernel developed

at MIT.

A review of GNU Project "GNUsletters" of the late 1980s reflects the management tension. In January, 1987, Stallman announced to the world that the GNU Project was working to overhaul TRIX, a Unix kernel developed at MIT. A year later, in February of 1988, the GNU Project announced that it had shifted its attentions to Mach, a lightweight "micro-kernel" developed at Carnegie Mellon. All told, however, official GNU Project kernel development wouldn't commence until 1990. [9]

The delays in kernel development were just one of many concerns weighing on Stallman during this period. In 1989, Lotus Development Corporation filed suit against rival software company, Paperback Software International, for copying menu commands in Lotus' popular 1-2-3 Spreadsheet program. Lotus' suit, coupled with the Apple-Microsoft "look and feel" battle, provided a troublesome backdrop for the GNU Project. Although both suits fell outside the scope of the GNU Project, both revolved around operating systems and software applications developed for the personal computer, not Unix-compatible hardware systems--they threatened to impose a chilling effect on the entire culture of software development. Determined to do something, Stallman recruited a few programmer friends and composed a magazine ad blasting the lawsuits. He then followed up the ad by helping to organize a group to protest the corporations filing the suit. Calling itself the League of Programming Freedom, the group held protests outside the offices of Lotus, Inc. and the Boston courtroom hosting the Lotus trial.

The protests were notable. [10] They document the evolving nature of software industry. Applications had quietly replaced operating systems as the primary corporate battleground. In its unfulfilled quest to build a free software operating system, the GNU Project seemed hopelessly behind the times. Indeed, the very fact that Stallman had felt it necessary to put together an entirely new group dedicated to battling the "look and feel" lawsuits reinforced that obsolescence in the eyes of some observers.

In 1990, the John D. and Catherine T. MacArthur Foundation cerified Stallman's genius status when it granted Stallman a MacArthur fellowship, therefore making him a recipient for the organization's so-called "genius grant." The grant, a $240,000 reward for launching the GNU Project and giving voice to the free software philosophy, relieved a number of short-term concerns. First and foremost, it gave Stallman, a nonsalaried employee of the FSF who had been supporting himself through consulting contracts, the ability to devote more time to writing GNU code. [11]

Ironically, the award also made it possible for Stallman to vote. Months before the award, a fire in Stallman's apartment house had consumed his few earthly possessions. By the time of the award, Stallman was listing himself as a "squatter" [12] at 545 Technology Square. "[The registrar of voters] didn't want to accept that as my address," Stallman would later recall. "A newspaper article about the MacArthur grant said that and then they let me register." [13]

Most importantly, the MacArthur money gave Stallman more freedom. Already dedicated to the issue of software freedom, Stallman chose to use the additional freedom to increase his travels in support of the GNU Project mission.

Interestingly, the ultimate success of the GNU Project and the free software movement in general would stem from one of these trips. In 1990, Stallman paid a visit to the Polytechnic University in Helsinki, Finland. Among the audience members was 21-year-old Linus Torvalds, future developer of the Linux kernel-the free software kernel destined to fill the GNU Project's most sizable gap.

A student at the nearby University of Helsinki at the time, Torvalds regarded Stallman with bemusement. "I saw, for the first time in my life, the stereotypical long-haired, bearded hacker type," recalls Torvalds in his 2001 autobiography [Just for Fun]. "We don’t have much of them in Helsinki." [14]

While not exactly attuned to the "sociopolitical" side of the Stallman agenda, Torvalds nevertheless appreciated the agenda’s underlying logic: no programmer writes error-free code. By sharing software, hackers put a program’s improvement ahead of individual motivations such as greed or ego protection.

Like many programmers of his generation, Torvalds had cut his teeth not on mainframe computers like the IBM 7094, but on a motley assortment of home-built computer systems. As university student, Torvalds had made the step up from C programming to Unix, using the university’s MicroVAX. This ladder-like progression had given Torvalds a different perspective on the barriers to machine access. For Stallman, the chief barriers were bureaucracy and privilege. For Torvalds, the chief barriers were geography and the harsh Helsinki winter. Forced to trek across the University of Helsinki just to log in to his Unix account, Torvalds quickly began looking for a way to log in from the warm confines of his off-campus apartment.

The search led Torvalds to the operating system Minix, a lightweight version of Unix developed for instructional purposes by Dutch university professor Andrew Tanenbaum. The program fit within the memory confines of a 386 PC, the most powerful machine Torvalds could afford, but still lacked a few necessary features. It most notably lacked terminal emulation, the feature that allowed Torvalds’ machine to mimic a university terminal, making it possible to log in to the MicroVAX from home.

During the summer of 1991, Torvalds rewrote Minix from the ground up, adding other features as he did so. By the end of the summer, Torvalds was referring to his evolving work as the "GNU/Emacs of terminal emulation programs." [15] Feeling confident, he solicited a Minix newsgroup for copies of the POSIX standards, the software blue prints that determined whether a program was Unix compatible. A few weeks later, Torvalds was posting a message eerily reminiscent of Stallman’s original 1983 GNU posting:

```
Hello everybody out there using minix-

    I’m doing a (free) operating system (just a hobby, won’t be big
    and professional like gnu for 386 (486) AT clones). This has been
    brewing since April, and is starting to get ready. I’d like any
    feedback on things people like/dislike in minix, as my OS
    resembles it somewhat (same physical layout of the file-system
    (due to practical reasons) among other things).
[16]
```

The posting drew a smattering of responses and within a month, Torvalds had posted a 0.01 version of the operating system--i.e., the earliest possible version fit for outside review--on an Internet FTP site. In the course of doing so, Torvalds had to come up with a name for the new system. On his own PC hard drive, Torvalds had saved the program as Linux, a name that paid its respects to the software convention of giving each Unix variant a name that ended with the letter X. Deeming the name too "egotistical," Torvalds changed it to Freax, only to have the FTP site manager change it back.

Although Torvalds had set out build a full operating system, both he and other developers knew at the time that most of the functional tools needed to do so were already available, thanks to the work of GNU, BSD, and other free software developers. One of the first tools the Linux development team took advantage of was the GNU C Compiler, a tool that made it possible to process programs written

in the C programming language.

Integrating GCC improved the performance of Linux. It also raised issues. Although the GPL's "viral" powers didn't apply to the Linux kernel, Torvald's willingness to borrow GCC for the purposes of his own free software operating system indicated a certain obligation to let other users borrow back. As Torvalds would later put it: "I had hoisted myself up on the shoulders of giants." [17] Not surprisingly, he began to think about what would happen when other people looked to him for similar support. A decade after the decision, Torvalds echoes the Free Software Foundation's Robert Chassel when he sums up his thoughts at the time:

> You put six months of your life into this thing and you want to make it available and you want to get something out of it, but you don't want people to take advantage of it. I wanted people to be able to see [Linux], and to make changes and improvements to their hearts' content. But I also wanted to make sure that what I got out of it was to see what they were doing. I wanted to always have access to the sources so that if they made improvements, I could make those improvements myself. [18]

When it was time to release the 0.12 version of Linux, the first to include a fully integrated version of GCC, Torvalds decided to voice his allegiance with the free software movement. He discarded the old kernel license and replaced it with the GPL. The decision triggered a porting spree, as Torvalds and his collaborators looked to other GNU programs to fold into the growing Linux stew. Within three years, Linux developers were offering their first production release, Linux 1.0, including fully modified versions of GCC, GDB, and a host of BSD tools.

By 1994, the amalgamated operating system had earned enough respect in the hacker world to make some observers wonder if Torvalds hadn't given away the farm by switching to the GPL in the project's initial months. In the first issue of Linux Journal, publisher Robert Young sat down with Torvalds for an interview. When Young asked the Finnish programmer if he felt regret at giving up private ownership of the Linux source code, Torvalds said no. "Even with 20/20 hindsight," Torvalds said, he considered the GPL "one of the very best design decisions" made during the early stages of the Linux project. [19]

That the decision had been made with zero appeal or deference to Stallman and the Free Software Foundation speaks to the GPL's growing portability. Although it would take a few years to be recognized by Stallman, the explosiveness of Linux development conjured flashbacks of Emacs. This time around, however, the innovation triggering the explosion wasn't a software hack like Control-R but the novelty of running a Unix-like system on the PC architecture. The motives may have been different, but the end result certainly fit the ethical specifications: a fully functional operating system composed entirely of free software.

As his initial email message to the comp.os.minix newsgroup indicates, it would take a few months before Torvalds saw Linux as anything less than a holdover until the GNU developers delivered on the HURD kernel. This initial unwillingness to see Linux in political terms would represent a major blow to the Free Software Foundation.

As far as Torvalds was concerned, he was simply the latest in a long line of kids taking apart and reassembling things just for fun. Nevertheless, when summing up the runaway success of a project that could have just as easily spent the rest of its days on an abandoned computer hard drive, Torvalds credits his younger self for having the wisdom to give up control and accept the GPL bargain.

"I may not have seen the light," writes Torvalds, reflecting on Stallman's 1991 Polytechnic University speech and his subsequent decision to switch to the GPL. "But I guess something from his speech sunk in." [20]

Notas

[1] Vea Hal Abelson, Mike Fischer, y Joanne Costello, "Software and Copyright Law", versión actualizada (1998). http://www.swiss.ai.mit.edu/6805/articles/int-prop/software-copyright.html

[2] See Trn Kit README. http://www.za.debian.org/doc/trn/trn-readme

[3] See John Gilmore, quoted from email to author.

[4] See Richard Stallman, et al., ‘‘GNU General Public License: Version 1,’’ (February, 1989). http://www.gnu.org/copyleft/copying-1.0.html

[5] See David Betz and Jon Edwards, "Richard Stallman discusses his public-domain [sic] Unix-compatible software system with BYTE editors," [BYTE] (July, 1996). (Reprinted on the GNU Project web site: http://www.gnu.org/gnu/byte-interview.html.) This interview offers an interesting, not to mention candid, glimpse at Stallman's political attitudes during the earliest days of the GNU Project. It is also helpful in tracing the evolution of Stallman's rhetoric. Describing the purpose of the GPL, Stallman says, "I'm trying to change the way people approach knowledge and information in general. I think that to try to own knowledge, to try to control whether people are allowed to use it, or to try to stop other people from sharing it, is sabotage." Contrast this with a statement to the author in August 2000: "I urge you not to use the term 'intellectual property' in your thinking. It will lead you to misunderstand things, because that term generalizes about copyrights, patents, and trademarks. And those things are so different in their effects that it is entirely foolish to try to talk about them at once. If you hear somebody saying something about intellectual property, without quotes, then he's not thinking very clearly and you shouldn't join."

[6] The University of California's "obnoxious advertising clause" would later prove to be a problem. Looking for a less restrictive alternative to the GPL, some hackers used the University of California, replacing "University of California" with the name of their own instution. The result: free software programs that borrowed from dozens of other programs would have to cite dozens of institutions in advertisements. In 1999, after a decade of lobbying on Stallman's part, the University of California agreed to drop this clause. See "The BSD License Problem" at http://www.gnu.org/philosophy/bsd.html.

[7] See Michael Tiemann, "Future of Cygnus Solutions: An Entrepreneur's Account," [Open Sources] (O'Reilly & Associates, Inc., 1999): 139.

[8] See Richard Stallman, [BYTE] (1986).

[9] See "HURD History." http://www.gnu.org/software/hurd/history.html

[10] According to a League of Programming Freedom Press, the protests were notable for featuring the first hexadecimal protest chant:

```
1-2-3-4, toss the lawyers out the door; 5-6-7-8, innovate don't
litigate; 9-A-B-C, 1-2-3 is not for me; D-E-F-O, look and feel
have got to go
```

http://lpf.ai.mit.edu/Links/prep.ai.mit.edu/demo.final.release

[11] I use the term "writing" here loosely. About the time of the MacArthur award, Stallman began suffering chronic pain in his hands and was dictating his work to FSF-employed typists. Although some have speculated that the hand pain was the result of repetitive stress injury, or RSI, an injury common among software programmers, Stallman is not 100% sure. "It was NOT carpal tunnel syndrome," he writes. "My hand problem was in the hands themselves, not in the wrists." Stallman has since learned to work without typists after switching to a keyboard with a lighter touch.

[12] See Reuven Lerner, "Stallman wins $240,000 MacArthur award," MIT, [The Tech] (July 18, 1990). http://the-tech.mit.edu/V110/N30/rms.30n.html

[13] See Michael Gross, "Richard Stallman: High School Misfit, Symbol of Free Software, MacArthur-certified Genius" (1999).

[14] See Linus Torvalds and David Diamond, [Just For Fun: The Story of an Accidentaly Revolutionary] (HarperCollins Publishers, Inc., 2001): 58-59.

[15] See Linus Torvalds and David Diamond, [Just For Fun: The Story of an Accidentaly Revolutionary] (HarperCollins Publishers, Inc., 2001): 78.

[16] See ‘‘Linux 10th Anniversary.’’ http://www.linux10.org/history/

[17] See Linus Torvalds and David Diamond, [Just For Fun: The Story of an Accidentaly Revolutionary] (HarperCollins Publishers, Inc., 2001): 96-97.

[18] See Linus Torvalds and David Diamond, [Just For Fun: The Story of an Accidentaly Revolutionary] (HarperCollins Publishers, Inc., 2001): 94-95.

[19] See Robert Young, "Interview with Linus, the Author of Linux," [Linux Journal] (March 1, 1994). http://www.linuxjournal.com/article.php?sid=2736

[20] See Linus Torvalds and David Diamond, [Just For Fun: The Story of an Accidentaly Revolutionary] (HarperCollins Publishers, Inc., 2001): 59.

Capítulo 10. GNU/Linux

Para 1993, el movimiento del software libre se encontraba en un cruce de caminos. Para los optimistas, todos los signos apuntaban hacía el éxito de la cultura hacker. La revista [Wired], una publicación nueva, con historias sobre encripción de datos, Usenet y libertad se software estaba desapareciendo de los estantes en las tiendas de revistas. Internet, alguna vez un término de la jerga de los hackers y los investigadores científicos, se había abierto camino y ya formaba parte del léxico de la gente común. Incluso el presidente Clinton estaba utilizando el término. El computador personal, que alguna vez fue un juguete de aficionados, había obtenido respeto en todos los estamentos, dándole a toda una nueva generación de usuarios acceso a software construido por hackers. Aunque el Proyecto GNU no había alcanzado aún su objetivo de un sistema operativo gratis completamente intacto, los usuarios curiosos podían probar Linux mientras que esto ocurría.

De cualquier manera que se vieran, las noticias eran buenas, o por lo menos así parecía. Después de una década de lucha, los hackers y sus valores finalmente estaban ganando aceptación dentro de la sociedad. Finalmente, el mensaje estaba llegándole a la gente.

¿Realmente lo estaba haciendo? Para los pesimistas, cada señal de aceptación incluía su propia señal contraria. Claro, ser un hacker estaba de repente de moda, pero ¿Era bueno estar de moda en una comunidad que siempre había crecido en la alienación? Cierto, la Casa Blanca estaba diciendo las cosas correctas respecto al Internet, llegando hasta el punto de registrar su propio nombre de dominio, *whitehouse.gov*, pero también estaba reuniéndose con las compañías, los representantes de la censura y los oficiales de las entidades policivas que intentaban poner freno a la cultura del salvaje oeste que reinaba en Internet. Seguro, los computadores personales eran poderosos, pero en el proceso de convertir a los PCs en un bien de consumo con sus chips, Intel había creado una situación en la que los vendedores de software propietario detentaban el poder. Por cada nuevo usuario ganado para la causa del software libre gracias a Linux cientos, quizás miles, estaban utilizando *Microsoft Windows* por primera vez.

Finalmente, era necesario tener en cuenta la naturaleza curiosa de Linux mismo. Sin restricciones de diseño (como GNU) y disputas legales (como BSD), la evolución a toda velocidad de Linux había sido tan poco planificada, y su éxito tan fortuito, que los programadores más cercanos al mismo código del software no sabían que hacer con él. Mas un disco de recopilaciones que un sistema operativo, estaba compuesto de una mezcla hacker de grandes éxitos: todo desde GCC, GDB y glibc (la librería de C recién desarrollada por el Proyecto GNU) pasando por X (una interfaz gráfica para el usuario basada en Unix y desarrollada por el Laboratorio de Ciencias de la Computación de MIT) y herramientas desarrolladas por BSD como BIND (el Demonio de Nombres de Berkeley para Internet o *Berkeley Internet Naming Daemon*, que permite a los usuarios utilizar nombres de dominio de Internet sencillos de recordar, como sustituto de las direcciones IP) y TCP/IP. La piedra angular de los arquitectos, por supuesto, era el mismo kernel de Linux. Una versión arreglada y supercargada de Minix. En lugar de construir su sistema operativo de la nada, Torvalds y el siempre creciente equipo de desarrollo de Linux habían seguido el adagio de Picasso, "Los buenos artistas toman prestado, los grandes artistas roban" O, como Torvalds mismo lo diría mas tarde, al describir el secreto de su éxito: "Yo soy básicamente una persona muy perezosa a la que le gusta llevarse el crédito por cosas que otra gente hace realmente." [1]

Esta pereza era tan admirable desde el punto de vista de la eficiencia, como problemática a nivel político. Con ella quedaba demostrada la carencia de una agenda ideológica de parte de Torvalds. A diferencia de los desarrolladores de GNU, Torvalds no había construido el sistema operativo porqué deseara darle a sus compañeros hackers algo para trabajar; Lo había hecho para poder tener algo con lo que él mismo pudiera jugar. El genio de Torvalds estaba menos en su visión de la estructura y más en su capacidad para reclutar otros hackers para acelerar el proceso.

Que Torvalds y sus reclutas hayan tenido éxito donde otros no lo habían echo trae su propia pregunta intrigante: ¿Qué era, exactamente, Linux? ¿Era una manifestación de la filosofía de software libre articulada por primera vez por Stallman en el Manifiesto GNU? ¿O era simplemente una agregación de herramientas de software interesantes que cualquier usuario, igualmente motivado, podría crear en la máquina de su casa?

Para finales de 1993, un número creciente de usuarios de Linux habían comenzado a inclinarse por la segunda definición, creando variaciones privadas sobre el tema de Linux. Llegaron hasta el punto de embotellar y vender sus variaciones-- o "distribuciones"-- a sus compañeros aficionados a Linux. Los resultados fueron mediocres, en el mejor de los casos.

"Todo esto sucedía antes de que existieran Red Hat y las otras distribuciones comerciales," recuerda Ian Murdock, quién en ese entonces era un estudiante de ciencias de la computación en la Universidad de Purdue. "Hojeando en las revistas de Unix era posible encontrar muchos avisos del tamaño de una tarjeta de presentación que proclamaban ‘Linux’. La mayoría de estas compañías eran operaciones nocturnas que no veían nada malo en agregarle su propio código fuente a la mezcla.

Murdock, un programador de Unix recuerda haberse "dejado llevar" por Linux cuando lo consiguió e instaló en el computador de su casa por primera vez. "Era simplemente una cosa muy divertida", dice. "Hacía que yo quisiera involucrarme". A pesar de eso, la explosión de distribuciones pobremente construidas comenzó a hacer mella en su entusiasmo inicial. Tras haber decidido que la mejor forma de involucrarse era construir una versión de Linux libre de aditivos Murdock comenzó a elaborar una lista con las mejores herramientas de software gratis disponible, con la intención de agruparlas en su propia distribución. "Quería algo que estuviera a la altura del nombre de Linux", dice Murdock.

En una intento por generar interés Murdock publicó sus intenciones en Internet, incluyendo el grupo de noticias de Usenet comp.os.linux. Uno de los primeros mensajes de respuesta provenía de mailto:rms@ai.mit.edu. Como hacker que era, Murdock reconoció instantáneamente la dirección. Era Richard M. Stallman, fundador del Proyecto GNU y un hombre que Murdock conocía incluso entonces como "el hacker de los hackers". Viendo esta dirección en su correo entrante, Murdock estaba desconcertado. ¿Por qué estaría alguién como Stallman, que dirigía su propio proyecto de sistema operativo, interesado en las gestiones de Murdock en el mundo Linux?

Murdock abrió el mensaje.

"El decía que la Fundación para el Software Libre estaba empezando a observar de cerca a Linux y que la FSF estaba también interesada, posiblemente, en hacer un sistema Linux. Básicamente, a Stallman le parecía que nuestros objetivos estaban alineados con su filosofía."

El mensaje representaba un cambio dramático para Stallman. Hasta 1993, Stallman se había conformado con mantenerse apartado de los asuntos de la comunidad Linux. De hecho, casi había ignorado el sistema operativo renegado cuando apareció por primera vez en el entorno de los programadores de Unix. Después de recibir la primera noticia de un sistema parecido a Unix que corría en computadores personales, Stallman dice que delegó la tarea de examinar el nuevo sistema operativo a un amigo; como lo recuerda Stallman, "El reportó que el sistema se había modelado

siguiendo a *System V*, que era la versión inferior de Unix. También me dijo que no era portable."

El reporte del amigo de Stallman era correcto. Creado para ejecutarse en máquinas basadas en el 386 de Intel, Linux estaba firmemente arraigado a esta plataforma de hardware de bajo costo. Lo que el amigo no reportó, sin embargo, fue la enorme ventaja de la que gozaba Linux como el único sistema operativo libremente modificable del mercado. En otras palabras, mientras que Stallman empleó los tres años siguientes ocupado en los reportes de errores de su equipo de HURD, Torvalds se ganaba a los programadores que mas tarde tomarían el sistema operativo para llevarlo a otras plataformas.

Para 1993, la inhabilidad del Proyecto GNU para despachar un kernel de sistema operativo que funcionará estaba generando problemas tanto dentro del Proyecto GNU mismo como en el movimiento de software libre como un todo. En marzo de 1993 un artículo de de la revista [Wired] escrito por Simson Garfinkel se refería al GNU como un proyecto "estancado" a pesar del éxito de la gran cantidad de herramientas del proyecto. [2] Aquellos que hacían parte del proyecto y de su organización sin ánimo de lucro, la Fundación para el Software Libre, recuerdan que los ánimos estaban aún peor de lo que hacía entrever el artículo. "Era muy claro, por lo menos para mi en ese momento, que había una ventana de oportunidad para introducir un nuevo sistema operativo", dice Chassell. "Y una vez esa ventana se cerrará, la gente se interesaría mucho menos. Que es exactamente lo que sucedió". [3]

Mucho se ha dicho acerca de los problemas del Proyecto GNU durante el periodo que va de 1990 a 1993. Aunque muchos culpan a Stallman por esos problemas, Eric Raymond, uno de los primeros miembros del equipo de GNU Emacs, y más tarde un crítico de Stallman dice que el problema fue, en su mayoría institucional. "La FSF se volvió arrogante", dice Raymond. "Se alejaron del de sistemas operativos." Aún peor, "Ellos pensaban que nada fuera de la FSF podría afectarlos."

Murdock, una persona menos involucrada con los asuntos internos del Proyecto GNU, ve las cosas de una manera más condescenciente. "Creo que parte del problema es que fueron demasiado ambiciosos y malgastaron su dinero", dice. "Los micro-kernels eran el tema de moda a finales de los 80 y principios de los 90. Desafortunadamente, ese fue el momento en el que el proyecto GNU comenzó a diseñar su kernel. Al final tenían gran cantidad de complicaciones innecesarias y habría sido necesario devolverse demasiado para deshacerse de ellas."

Stallman cita varias circunstancias cuando explica la demora. Las demandas de Lotus y Apple habían creado distracciones políticas que, sumadas con la incapacidad de Stallman para teclear, hacían difícil para él darle una mano al equipo de HURD. Stallman también habla de una falta de comunicación entre los diversos componentes del Proyecto GNU. "Tuvimos que trabajar mucho para que el ambiente de depuración funcionara", recuerda. "Y la gente encargada del mantenimiento de GDB en ese momento no cooperaba demasiado". Sin embargo Stallman afirma que la mayoría de los retrasos se debieron a la posición de Stallman y otros miembros del equipo de GNU, que subestimaron la dificultad involucrada en convertir el micro kernel de Mach en un kernel de Unix completo.

"Yo pensé: bueno, la parte [de Mach] que tiene que hablar con la máquina ya fue depurada", dice Stallman, recordando los problemas del equipo HURD en un discurso. "Con esa ventaja, deberíamos ser capaces de lograrlo más rápido. Pero en lugar de eso, resultó que depurar estos programas asíncronos con varios hilos era muy difícil. Habían *timing books that would clobber the files*, y eso no tiene nada de gracioso. El resultado final fue que se necesitaron muchos, muchos años para producir una versión de prueba." [4]

Cualquiera que fuera la excusa, o excusas, el éxito simultáneo del equipo del kernel de Linux creó una situación muy tensa. Seguro, el kernel de Linux había sido licenciado con la licencia GPL, pero, como Murdock mismo lo había dicho, el deseo de tratar a Linux como un sistema operativo puramente libre distaba mucho de ser unánime. Para finales de 1993, la población total de usuarios de Linux había crecido de una docena de entusiastas de Minix a entre 20,000 y 100,000 usuarios. [5] Lo que alguna vez había sido un pasatiempo era ahora un mercado maduro para ser explotado. Como Winston Churchill al ver a las tropas soviéticas tomándose Berlin, Stallman sentía una mezcla, bastante entendible, de emociones contradictorias en el momento de celebrar la "victoria" de Linux. [6]

A pesar de llegar tarde a la fiesta, Stallman aún tenía poder. Tan pronto como la FSF anunció que le daría su dinero y su soporte moral al proyecto de software de Murdock, otras ofertas de ayuda empezaron a llegar, Murdock llamó al nuevo proyecto Debian-- una compresión de su propio nombre y el de su esposa, Deborah -- y unas semanas más tarde estaba entregando la primera distribución. "La ayuda de Richard catapultó a Debian casi de un día para otro de un pequeño proyecto interesante, para convertirlo en algo a lo que la gente de la comunidad debía prestarle atención" dice Murdock.

En enero de 1994, Murdock hizo público el "Manifiesto Debian". Escrito con el espíritu del "Manifiesto GNU" de Stallman, que le precedía diez años, explicaba la importancia de trabajar de manera muy cercana con la Fundación para el Software Libre. Murdock escribió:

> La Fundación pare el Software Libre tiene un papel extremadamente importante en el futuro de Debian. Por el simple hecho de ser ellos quienes lo distribuyen, se envía un mensaje al mundo; se le dice que Linux no es un producto comercial y que nunca debería serlo, sin que esto signifique que Linux nunca será capaz de competir comercialmente. Para aquellos de ustedes que esten en desacuerdo, los reto a racionalizar el éxito de GNU Emacs y GCC, que no son software comercial, pero que a pesar de ello han tenido un impacto importante en el mercado comercial. Ha llegado la hora de concentrarse en el futuro de Linux, en lugar de buscar el objetivo destructivo de enriquecerse a uno mismo a costa de la comunidad de Linux y de su futuro. El desarrollo y distribución de Debian pueden no ser la solución a los problemas que he descrito en el Manifiesto, pero espero que por lo menos atraigan la suficiente atención a estos problemas como para permitir que sean resueltos. [7]

Poco después de la publicación del Manifiesto la Fundación para el Software Libre hizo su primera solicitud importante. Stallman quería que Murdock llamará a su distribución "GNU/Linux". En un principio, dice Murdock, Stallman quería usar el término "Lignux", para decir "Linux con GNU en el corazón"-- pero una prueba del término en Usenet y en varios grupos de hackers había motivado la suficiente cantidad de mociones de desaprobación como para convencer a Stallman de inclinarse por el menos extravagante GNU/Linux.

A pesar de que algunos tomarían el intento de Stallman de adicionar el prefijo "GNU" como una búsqueda tardía de crédito, Murdock lo percibió de manera diferente. En perspectiva, Murdock lo vió como un intento para contrarrestar la creciente tensión entre el Proyecto GNU y los desarrolladores del kernel de Linux. "Había una separación en ciernes", recuerda Murdock. "Richard estaba preocupado."

La diferencia más profunda, según Murdock, se refería a glibc. Abreviatura para *GNU C Library* (Librería de C de GNU), glibc es el paquete que les permite a los programadores hacer "llamadas del sistema" dirigidas hacía el kernel. Durante los años 1993-1994, glibc se convirtió en un problemático cuello de botella en el desarrollo de Linux. Como había muchos nuevos usuarios adicionando funciones al kernel de Linux los encargados de mantener glibc en el Proyecto GNU pronto se vieron superados por la cantidad de cambios sugeridos. Frustrados por las demoras y por la creciente reputación del Proyecto GNU de "arrastrar los pies", algunos de los desarrolladores sugirieron crear

una bifurcación, es decir, una Librería de C específica a Linux y paralela a glibc.

En el mundo de los hackers los llamados *"forks"* o bifurcaciones son un fenómeno interesante. A pesar de que la ética hacker le permite a un programador hacer lo que desee con el código fuente de un programa cualquiera, la mayoría de los hackers prefieren vertir sus innovaciones en un archivo central de código fuente, o árbol, para asegurar la compatibilidad con los programas de otras personas. Bifurcar glibc en un estado tan temprano del desarrollo de Linux habría significado perder los aportes de cientos, si no miles, de desarrolladores de Linux. También habría significado una incompatibilidad creciente entre Linux y el sistema GNU que Stallman y el equipo GNU aún esperaban desarrollar.

Como líder del proyecto GNU, Stallman ya había experimentado los efectos negativos de una bifurcación del código en 1991. Un grupo de desarrolladores de Emacs trabajando para una compañía de software llamada Lucid habían discutido con Stallman cuando este no había incluido los cambios de vuelta en el código principal de Emacs. La bifurcación había dado origen a una versión paralela, Lucid Emacs, y había resentimiento en el ambiente. [8]

Murdock dice que Debian estaba comenzando a trabajar en una bifurcación similar en el código fuente de glibc que motivó a Stallman a insistir en adicionar el prefijo GNU cuando Debian entregó su distribución de software. "La bifurcación ha vuelto a converger desde entonces. Sin embargo, en ese momento había preocupación porqué si la comunidad Linux se veía a si misma como algo distinto de la comunidad GNU entonces habría una fuerza que podría llevar a la separación."

Stallman esta de acuerdo con la forma de ver los hechos de Murdock. De hecho, el dice que habían grandes bifurcaciones en ciernes en cada uno de los proyectos importantes de GNU. En un principio, Stallman dice que consideró las bifuraciones como el producto de algunos miembros inconformes. A diferencia de la dinámica veloz e informal del equipo del kernel de Linux, los encargados de mantenimiento del código fuente de GNU tendían a ser mas lentos y mas reflexivos al hacer cambios que pudieran afectar la viabilidad a largo plazo de un programa. Además, no tenían miedo de criticar duramente el código de otras personas. Con el paso del tiempo, Stallman comenzó a sentir que había una falta de conciencia latente sobre el proyecto GNU y sus objetivos cuando leía los correos de los desarrolladores de Linux.

"Descubrimos que a las personas que se consideraban a si mismas como usuarios de Linux no les importaba el Proyecto GNU", dice Stallman. "Ellos decían: ‘¿Por qué debería preocuparme haciendo estas cosas? No me importa el Proyecto GNU. Las cosas funcionan para mí. Las cosas funcionan para nosotros, los usuarios de Linux, y nada mas nos importa.’ Este fenómeno era bastante sorprendente, ya que la gente estaba esencialmente usando una variante del sistema GNU, y les importaba muy poco. Les importaba GNU menos que a cualquier otro."

Mientras que algunos veían las descripciones de Linux como una "variante" del proyecto GNU como políticamente tentadoras, Murdock, que ya simpatizaba con la causa del software libre, vió la solicitud de Stallman para llamar a la versión de Debian GNU/Linux como algo razonable. "Era más por la unidad que por quedarse con el crédito", dice.

Solicitudes de una naturaleza más técnica inmediatamente se hicieron presentes. A pesar de que Murdock se había adaptado en los temas políticos, tuvo una posición mucho más firme en cuanto se refiere al diseño y al modelo del desarrollo del software como tal. Lo que había comenzado como una muestra de solidaridad pronto se convirtió en un modelo para otros proyectos de GNU.

"Puedo decirte que he tenido varios desacuerdos con el," dice Murdock con una sonrisa. "Siendo honesto, Richard es a veces una persona con la que es bastante difícil trabajar."

En 1996, Murdock, tras graduarse de Purdue decidió entregar las riendas del creciente proyecto Debian. Ya había estado cediendo tareas de gerencia a Bruce Perens, el hacker mejor conocido por su trabajo en Electric Fence, una utilidad de Unix distribuida bajo la GPL. Perens, como Murdock, era un programador de Unix que se había enamorado de Linux tan pronto como las habilidades Unix del programa se hicieron manifiestas. Al igual que Murdock, Perens se identificaba con los ideales políticos de Stallman y la Fundación para el Software Libre, aunque desde lejos.

"Recuerdo que luego de que Stallman ya había hecho el Manifiesto GNU, GNU Emacs y GCC, leí un artículo en el que se decía que el estaba trabajando para Intel cmo consultor," dice Perens, recordando su primer contacto con Stallman, a finales de los ochentas. "Le escribí preguntándole como podía estar apoyando el software libre con una mano y trabajando para Intel con la otra. El devolvió el mensaje diciendo, 'Yo trabajo como consultor para crear software libre.' Fue tolatmente cortes al respecto, y yo pensé que su respuesta tenía mucho sentido."

Como un desarrollador prominente de Debian, sin embargo Perens veía las batallas de diseño de Murdock con Stallman con desgano. Al asumir el liderazgo del equipo de desarrollo, Perens dice que tomó la decisión de distanciar a Debian de la Fundación para el Software Libre. "Decidí que no queríamos el estilo de micro-gerencia de Richard", dice.

Según Perens, Stallman estaba sorprendido con la noticia pero tuvo la suficiente sabiduría como para aceptarla. "El le dió algo de tiempo para que se enfriara y envió un mensaje diciendo que realmente necesitábamos algún tipo de relación. Él solicitó que lo llamaramos GNU/Linux y lo dejáramos así. Yo decidí que estaba bien. Tomé esa decisión de manera unilateral, todos suspiraron aliviados."

Con el tiempo, Debian se formaría una reputación como la versión de Linux de los hackers, junto a Slackware, otra distribución popular creada durante el periodo 1993-1994. Fuera del reino de los sistemas orientados hacía los hackers, Linux estaba comenzando a ser noticia en el mercado de Unix Comercial. En Carolina del Norte una compañía que se llamaba a si misma Red Hat estaba reinventando su esquema de negocios para concentrarse en Linux. El CEO de esa compañía era Robert Young, el antiguo editor de [Linux Journal] que había preguntado en 1994 a Linus Torvalds si se arrepentía de haber distribuido el kernel de Linux con la licensia GPL. Para Young la respuesta de Torvalds tuvo un impacto "profundo" en su forma de ver Linux. En lugar de buscar una manera de arrinconar el mercado, por medio de tácticas de software tradicionales, Young comenzó a pensar que sucedería si una compañía hacía lo mismo que Debian --es decir, construir un sistema operativo basado completamente en partes de software libre. Cygnus Solutions, la compañía fundada por Michael Tiemann y John Gilmore en 1990, ya estaba demostrando su capacidad para vender software libre basándose en la calidad y en la flexibilidad. ¿Qué sucedería si Red Hat hacía lo mismo con GNU/Linux?

"En la tradición científica occidental nos apoyamos en los hombros de gigantes," dice Young, retomando tanto a Torvalds como a Sir Isaac Newton como ejemplo. "En los negocios, eso significa to tener que reinventar la rueda a medida que vamos andado. La belleza del modelo [GPL] es que cada uno hace que su código sea del dominio público. [9] Si usted es un vendedor de software independiente que esta tratando de crear una aplicación y necesita un marcador de modem, pues ¿Para que reinventar los marcadores de modem? Puede simplemente robarle PPP a Red Hat Linux y usar eso como la base para su herramienta de marcado de modem. Si necesita un conjunto de herramientas gráficas, usted no tiene que escribir su propia librería. Simplemente baje GTK. De repente usted tiene la posibilidad de reutilizar lo mejor de lo que se hizo antes. Y de repente su objetivo como vendedor de

aplicaciones se centra menos en la gestión del software y más en escribir aplicaciones específicas a las necesidades de sus clientes."

Young no era el único ejecutivo del software intrigado con las eficiencias de negocios del software libre. Para finales de 1996, la mayoría de las compañías de Unix estaban comenzando a darse cuenta de la existencia de todo ese código fuente. El sector de Linux aún estaba a un año o dos de su explosión comercial completa, pero aquellos lo suficientemente cercanos a la comunidad hacker lo pudieron sentir: algo grande estaba ocurriendo. El chip 386 de Intel, la Internet, y la web habían golpeado el mercado como un conjunto de olas gigantes, y Linix-- y el grupo de programas que lo acompañaban en términos de accesibilidad del código y permisividad de las licencias-- parecían ser la ola más grande hasta entonces.

Para Ian Murdock, el programador buscado por Stallman y después decepcionado por el estilo de microgerencia de este, la ola parecía tanto un tributo como un castigo justo para el hombre que había empleado tanto tiempo dándole al movimiento de software libre una identidad. Como muchos aficionados a Linux, Murdock había visto los anuncios originales. Había visto la propuesta inicial de Torvalds en la que decía que Linux era "solo un pasatiempo". También había sido testigo de la "confesión" de Torvalds frente al creador de Minix, Andrew Tannenbaum en la que decía "Si el kernel de GNU hubiera estado listo la primavera pasada, yo ni siquiera me habría molestado en iniciar mi proyecto." [10] Como muchos, Murdock sabía de las oportunidades que se habían desperdiciado. También conocía la emoción de ver las nuevas oportunidades emerger del tejido mismo de la Internet.

"Estar involucrado con Linux en esos primeros días fue muy divertido", recueda Murdock. "Al mismo tiempo, era algo para hacer, algo para pasar el tiempo. Si usted vuelve a leer aquellos viejos [comp.os.minix] intercambios, usted verá el sentimiento general: esto es algo con lo que podemos jugar hasta que HURD este listo. La gente estaba ansiosa. Es divertido pero, de muchas maneras, sospecho que Linux nunca habría ocurrido si HURD hubiera aparecido mas rápidamente."

Para finales de 1996, sin embargo, todas esa preguntas de "que pasaría si" ya estaban debatidas. Llámelo Linux, llámelo GNU/Linux; los usuarios ya habían hablado. La ventana de 36 meses se había cerrado, por lo que incluso si el Proyecto GNU hubiera publicado su kernel HURD, las probabilidades de que alguién fuera de la comunidad de los hackers más dedicados se hubiera dado cuenta, eran remotas. El primer sistema operativo libre parecido a Unix había llegado, y ya tenía momento. Todo lo que tenían que hacer los hackers era relajarse y esperar que la próxima gran ola viniera a estrellarse contra sus cabezas. Incluso en la revuelta cabellera de un tal Richard M. Stallman.

Listos o no.

Notas

[1] Torvalds ha dicho esta frase en entornos muy diferentes. A la fecha, sin embargo, la aparición más notable de esta cita se encuentra en el ensayo de Eric Raymond, "La Catedral y el Bazar" (Mayo de 1997) http://www.tuxedo.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/index.html

[2] Ver Simson Garfinkel, "Is Stallman Stalled?" [Wired] (Marzo, 1993).

[3] La preocupación de Chassell acerca de la existencia de una "ventana" de 36 meses para un nuevo sistema operativo no es exclusiva del Proyecto GNU. A comienzos de los noventa, versiones libres de la Distribución de Software de Berkley (BSD) estuvieron restringidas por la demanda de Unix System Laboratories que restringía la publicación de software derivado de BSD. Aún cuando muchos usuarios consideran a los subproductos de BSD, como FreeBSD y OpenBSD como sistemas demostrablemente superiores a GNU/Linux tanto en términos de desempeño como en términos de seguridad, el número de usuarios de ambos sistemas permanece como tan solo una fracción del total de usuarios de GNU/Linux. Para ver un análisis muestral del éxito relativo de GNU/Linux comparado con otros sistemas operativos libres, vea el ensayo del hacker neozelandés, Liam Greenwood *"Why is Linux Successful"*(¿Por qué Linux es exitoso?), publicado en 1999.

[4] Ver el discurso en el Centro de Computación de Alto Rendimiento de Maui.

[5] Los datos sobre la cantidad de usuarios de GNU/Linux son ,en el mejor de los casos, bastante inexactos. Por esos es que he mencionado un rango tan amplio. El total de 100,000 viene del sitio de "Marcas" de Red Hat, http://www.redhat.com/about/corporate/milestones.html.

[6] Escribí esta analogía con Winston Churchill antes de que Stallman mismo me enviara su propio comentario no solicitado sobre Churchill:

> La Segunda Guerra Mundial y la fuerza de voluntan necesaria para ganarla fueron un pensamiento muy fuerte durante mi crecimiento. Declaraciones como, "Pelearemos de igual manera en las playas como en tierra firme, en los campos y llanos como en las calles." siempre han estado presentes para mí.

[7] Ver Ian Murdock, "A Brief History of Debian" (Una Breve Historia de Debian), agosto 6 de 1994: Apéndice A, "The Debian Manifesto." http://www.debian.org/doc/manuals/project-history/apA.html

[8] Jamie Zawinski, un antiguo programador de Lucid, que después saldría a encabezar el equipo de desarrollo de Mozilla, tiene un sitio web que documenta la bifurcación Lucid/GNU Emacs, llamado *"The Lemacs/FSFmacs Schism."* http://www.jwz.org/doc/lemacs.html

[9] Young usa el término "dominio público" de manera incorrecta aquí. De dominio público significa que no esta protegido por derechos de autor. Los programas amparador por la GPL están, por definición, protegidos por derechos de autor.

[10] Esta cita fue tomada de la muy publicitada "guerra de insultos" (*flame war*) que siguió al lanzamiento inicial de Linux. En el proceso de defender su elección de un diseño monolítico y no portable dek kernelm, Torvalds dice que comenzó a trabajar en Linux como una manera de aprender más acerca de su nuevo PC 386. "Si el kernel de GNU hubiera estado listo la primavera pasada, yo ni siquiera me habría molestado en iniciar mi proyecto." Ver Chris DiBona et al., [Open Sources] (O'Reilly & Associates, Inc., 1999): 224.

Capítulo 11. Open Source

In November, 1995, Peter Salus, a member of the Free Software Foundation and author of the 1994 book, [A Quarter Century of Unix], issued a call for papers to members of the GNU Project's "system-discuss" mailing list. Salus, the conference's scheduled chairman, wanted to tip off fellow hackers about the upcoming Conference on Freely Redistributable Software in Cambridge, Massachusetts. Slated for February, 1996 and sponsored by the Free Software Foundation, the event promised to be the first engineering conference solely dedicated to free software and, in a show of unity with other free software programmers, welcomed papers on "any aspect of GNU, Linux, NetBSD, 386BSD, FreeBSD, Perl, Tcl/tk, and other tools for which the code is accessible and redistributable." Salus wrote:

> Over the past 15 years, free and low-cost software has become ubiquitous. This conference will bring together implementers of several different types of freely redistributable software and publishers of such software (on various media). There will be tutorials and refereed papers, as well as keynotes by Linus Torvalds and Richard Stallman. [1]

One of the first people to receive Salus' email was conference committee member Eric S. Raymond. Although not the leader of a project or company like the various other members of the list, Raymond had built a tidy reputation within the hacker community as a major contributor to GNU Emacs and as editor of [The New Hacker Dictionary], a book version of the hacking community's decade-old Jargon File.

For Raymond, the 1996 conference was a welcome event. Active in the GNU Project during the 1980s, Raymond had distanced himself from the project in 1992, citing, like many others before him, Stallman's "micro-management" style. "Richard kicked up a fuss about my making unauthorized modifications when I was cleaning up the Emacs LISP libraries," Raymond recalls. "It frustrated me so much that I decided I didn't want to work with him anymore."

Despite the falling out, Raymond remained active in the free software community. So much so that when Salus suggested a conference pairing Stallman and Torvalds as keynote speakers, Raymond eagerly seconded the idea. With Stallman representing the older, wiser contingent of ITS/Unix hackers and Torvalds representing the younger, more energetic crop of Linux hackers, the pairing indicated a symbolic show of unity that could only be beneficial, especially to ambitious younger (i.e., below 40) hackers such as Raymond. "I sort of had a foot in both camps," Raymond says.

By the time of the conference, the tension between those two camps had become palpable. Both groups had one thing in common, though: the conference was their first chance to meet the Finnish *wunderkind* in the flesh. Surprisingly, Torvalds proved himself to be a charming, affable speaker. Possessing only a slight Swedish accent, Torvalds surprised audience members with his quick, self-effacing wit. [2]

Even more surprising, says Raymond, was Torvalds' equal willingness to take potshots at other prominent hackers, including the most prominent hacker of all, Richard Stallman. By the end of the conference, Torvalds' half-hacker, half-slacker manner was winning over older and younger conference-goers alike.

"It was a pivotal moment," recalls Raymond. "Before 1996, Richard was the only credible claimant to being the ideological leader of the entire culture. People who dissented didn't do so in public. The person who broke that taboo was Torvalds."

The ultimate breach of taboo would come near the end of the show. During a discussion on the growing market dominance of Microsoft Windows or some similar topic, Torvalds admitted to being a fan of Microsoft's PowerPoint slideshow software program. From the perspective of old-line software purists, it was like a Mormon bragging in church about his fondness of whiskey. From the perspective of Torvalds and his growing band of followers, it was simply common sense. Why shun worthy proprietary software programs just to make a point? Being a hacker wasn't about suffering, it was about getting the job done.

"That was a pretty shocking thing to say," Raymond remembers. "Then again, he was able to do that, because by 1995 and 1996, he was rapidly acquiring clout."

Stallman, for his part, doesn't remember any tension at the 1996 conference, but he does remember later feeling the sting of Torvalds' celebrated cheekiness. "There was a thing in the Linux documentation which says print out the GNU coding standards and then tear them up," says Stallman, recalling one example. "OK, so he disagrees with some of our conventions. That's fine, but he picked a singularly nasty way of saying so. He could have just said 'Here's the way I think you should indent your code.' Fine. There should be no hostility there."

For Raymond, the warm reception other hackers gave to Torvalds' comments merely confirmed his suspicions. The dividing line separating Linux developers from GNU/Linux developers was largely generational. Many Linux hackers, like Torvalds, had grown up in a world of proprietary software. Unless a program was clearly inferior, most saw little reason to rail against a program on licensing issues alone. Somewhere in the universe of free software systems lurked a program that hackers might someday turn into a free software alternative to PowerPoint. Until then, why begrudge Microsoft the initiative of developing the program and reserving the rights to it?

As a former GNU Project member, Raymond sensed an added dynamic to the tension between Stallman and Torvalds. In the decade since launching the GNU Project, Stallman had built up a fearsome reputation as a programmer. He had also built up a reputation for intransigence both in terms of software design and people management. Shortly before the 1996 conference, the Free Software Foundation would experience a full-scale staff defection, blamed in large part on Stallman. Brian Youmans, a current FSF staffer hired by Salus in the wake of the resignations, recalls the scene: "At one point, Peter [Salus] was the only staff member working in the office."

For Raymond, the defection merely confirmed a growing suspicion: recent delays such as the HURD and recent troubles such as the Lucid-Emacs schism reflected problems normally associated with software project management, not software code development. Shortly after the Freely Redistributable Software Conference, Raymond began working on his own pet software project, a popmail utility called "fetchmail." Taking a cue from Torvalds, Raymond issued his program with a tacked-on promise to update the source code as early and as often as possible. When users began sending in bug reports and feature suggestions, Raymond, at first anticipating a tangled mess, found the resulting software surprisingly sturdy. Analyzing the success of the Torvalds approach, Raymond issued a quick analysis: using the Internet as his "petri dish" and the harsh scrutiny of the hacker community as a form of natural selection, Torvalds had created an evolutionary model free of central planning.

What’s more, Raymond decided, Torvalds had found a way around Brooks’ Law. First articulated by Fred P. Brooks, manager of IBM’s OS/360 project and author of the 1975 book, [The Mythical Man-Month], Brooks’ Law held that adding developers to a project only resulted in further project delays. Believing as most hackers that software, like soup, benefits from a limited number of cooks, Raymond sensed something revolutionary at work. In inviting more and more cooks into the kitchen, Torvalds had actually found away to make the resulting software *better*. [3]

Raymond put his observations on paper. He crafted them into a speech, which he promptly delivered before a group of friends and neighbors in Chester County, Pennsylvania. Dubbed "The Cathedral and the Bazaar," the speech contrasted the management styles of the GNU Project with the management style of Torvalds and the kernel hackers. Raymond says the response was enthusiastic, but not nearly as enthusiastic as the one he received during the 1997 Linux Kongress, a gathering of Linux users in Germany the next spring.

"At the Kongress, they gave me a standing ovation at the end of the speech," Raymond recalls. "I took that as significant for two reasons. For one thing, it meant they were excited by what they were hearing. For another thing, it meant they were excited even after hearing the speech delivered through a language barrier."

Eventually, Raymond would convert the speech into a paper, also titled "The Cathedral and the Bazaar." The paper drew its name from Raymond’s central analogy. GNU programs were "cathedrals," impressive, centrally planned monuments to the hacker ethic, built to stand the test of time. Linux, on the other hand, was more like "a great babbling bazaar," a software program developed through the loose decentralizing dynamics of the Internet.

Implicit within each analogy was a comparison of Stallman and Torvalds. Where Stallman served as the classic model of the cathedral architect--i.e., a programming "wizard" who could disappear for 18 months and return with something like the GNU C Compiler--Torvalds was more like a genial dinner-party host. In letting others lead the Linux design discussion and stepping in only when the entire table needed a referee, Torvalds had created a development model very much reflective of his own laid-back personality. From the Torvalds’ perspective, the most important managerial task was not imposing control but keeping the ideas flowing.

Summarized Raymond, "I think Linus’s cleverest and most consequential hack was not the construction of the Linux kernel itself, but rather his invention of the Linux development model." [4]

In summarizing the secrets of Torvalds’ managerial success, Raymond himself had pulled off a coup. One of the audience members at the Linux Kongress was Tim O’Reilly, publisher of O’Reilly & Associates, a company specializing in software manuals and software-related books (and the publisher of this book). After hearing Raymond’s Kongress speech, O’Reilly promptly invited Raymond to deliver it again at the company’s inaugural Perl Conference later that year in Monterey, California.

Although the conference was supposed to focus on Perl, a scripting language created by Unix hacker Larry Wall, O’Reilly assured Raymond that the conference would address other free software technologies. Given the growing commercial interest in Linux and Apache, a popular free software web server, O’Reilly hoped to use the event to publicize the role of free software in creating the entire infrastructure of the Internet. From web-friendly languages such as Perl and Python to back-room programs such as BIND (the Berkeley Internet Naming Daemon), a software tool that lets users replace arcane IP numbers with the easy-to-remember domain-name addresses (e.g., amazon.com), and sendmail, the most popular mail program on the Internet, free software had become an emergent phenomenon. Like a colony of ants creating a beautiful nest one grain of sand at a time, the only thing missing was the communal self-awareness. O’Reilly saw Raymond’s speech as a good way to inspire

that self-awareness, to drive home the point that free software development didn't start and end with the GNU Project. Programming languages, such as Perl and Python, and Internet software, such as BIND, sendmail, and Apache, demonstrated that free software was already ubiquitous and influential. He also assured Raymond an even warmer reception than the one at Linux Kongress.

O'Reilly was right. "This time, I got the standing ovation before the speech," says Raymond, laughing.

As predicted, the audience was stocked not only with hackers, but with other people interested in the growing power of the free software movement. One contingent included a group from Netscape, the Mountain View, California startup then nearing the end game of its three-year battle with Microsoft for control of the web-browser market.

Intrigued by Raymond's speech and anxious to win back lost market share, Netscape executives took the message back to corporate headquarters. A few months later, in January, 1998, the company announced its plan to publish the source code of its flagship Navigator web browser in the hopes of enlisting hacker support in future development.

When Netscape CEO Jim Barksdale cited Raymond's "Cathedral and the Bazaar" essay as a major influence upon the company's decision, the company instantly elevated Raymond to the level of hacker celebrity. Determined not to squander the opportunity, Raymond traveled west to deliver interviews, advise Netscape executives, and take part in the eventual party celebrating the publication of Netscape Navigator's source code. The code name for Navigator's source code was "Mozilla": a reference both to the program's gargantuan size--30 million lines of code--and to its heritage. Developed as a proprietary offshoot of Mosaic, the web browser created by Marc Andreessen at the University of Illinois, Mozilla was proof, yet again, that when it came to building new programs, most programmers preferred to borrow on older, modifiable programs.

While in California, Raymond also managed to squeeze in a visit to VA Research, a Santa Clara-based company selling workstations with the GNU/Linux operating system preinstalled. Convened by Raymond, the meeting was small. The invite list included VA founder Larry Augustin, a few VA employees, and Christine Peterson, president of the Foresight Institute, a Silicon Valley think tank specializing in nanotechnology.

"The meeting's agenda boiled down to one item: how to take advantage of Netscape's decision so that other companies might follow suit?" Raymond doesn't recall the conversation that took place, but he does remember the first complaint addressed. Despite the best efforts of Stallman and other hackers to remind people that the word "free" in free software stood for freedom and not price, the message still wasn't getting through. Most business executives, upon hearing the term for the first time, interpreted the word as synonymous with "zero cost," tuning out any follow up messages in short order. Until hackers found a way to get past this cognitive dissonance, the free software movement faced an uphill climb, even after Netscape.

Peterson, whose organization had taken an active interest in advancing the free software cause, offered an alternative: open source.

Looking back, Peterson says she came up with the open source term while discussing Netscape's decision with a friend in the public relations industry. She doesn't remember where she came upon the term or if she borrowed it from another field, but she does remember her friend disliking the term. [5]

At the meeting, Peterson says, the response was dramatically different. "I was hesitant about suggesting it," Peterson recalls. "I had no standing with the group, so started using it casually, not highlighting it as a new term." To Peterson's surprise, the term caught on. By the end of the meeting,

most of the attendees, including Raymond, seemed pleased by it.

Raymond says he didn't publicly use the term "open source" as a substitute for free software until a day or two after the Mozilla launch party, when O'Reilly had scheduled a meeting to talk about free software. Calling his meeting "the Freeware Summit," O'Reilly says he wanted to direct media and community attention to the other deserving projects that had also encouraged Netscape to release Mozilla. "All these guys had so much in common, and I was surprised they didn't all know each other," says O'Reilly. "I also wanted to let the world know just how great an impact the free software culture had already made. People were missing out on a large part of the free software tradition."

In putting together the invite list, however, O'Reilly made a decision that would have long-term political consequences. He decided to limit the list to west-coast developers such as Wall, Eric Allman, creator of sendmail, and Paul Vixie, creator of BIND. There were exceptions, of course: Pennsylvania-resident Raymond, who was already in town thanks to the Mozilla launch, earned an quick invite. So did Virginia-resident Guido van Rossum, creator of Python. "Frank Willison, my editor in chief and champion of Python within the company, invited him without first checking in with me," O'Reilly recalls. "I was happy to have him there, but when I started, it really was just a local gathering."

For some observers, the unwillingness to include Stallman's name on the list qualified as a snub. "I decided not to go to the event because of it," says Perens, remembering the summit. Raymond, who did go, says he argued for Stallman's inclusion to no avail. The snub rumor gained additional strength from the fact that O'Reilly, the event's host, had feuded publicly with Stallman over the issue of software-manual copyrights. Prior to the meeting, Stallman had argued that free software manuals should be as freely copyable and modifiable as free software programs. O'Reilly, meanwhile, argued that a value-added market for nonfree books increased the utility of free software by making it more accessible to a wider community. The two had also disputed the title of the event, with Stallman insisting on "Free Software" over the less politically laden "Freeware."

Looking back, O'Reilly doesn't see the decision to leave Stallman's name off the invite list as a snub. "At that time, I had never met Richard in person, but in our email interactions, he'd been inflexible and unwilling to engage in dialogue. I wanted to make sure the GNU tradition was represented at the meeting, so I invited John Gilmore and Michael Tiemann, whom I knew personally, and whom I knew were passionate about the value of the GPL but seemed more willing to engage in a frank back-and-forth about the strengths and weaknesses of the various free software projects and traditions. Given all the later brouhaha, I do wish I'd invited Richard as well, but I certainly don't think that my failure to do so should be interpreted as a lack of respect for the GNU Project or for Richard personally."

Snub or no snub, both O'Reilly and Raymond say the term "open source" won over just enough summit-goers to qualify as a success. The attendees shared ideas and experiences and brainstormed on how to improve free software's image. Of key concern was how to point out the successes of free software, particularly in the realm of Internet infrastructure, as opposed to playing up the GNU/Linux challenge to Microsoft Windows. But like the earlier meeting at VA, the discussion soon turned to the problems associated with the term "free software." O'Reilly, the summit host, remembers a particularly insightful comment from Torvalds, a summit attendee.

"Linus had just moved to Silicon Valley at that point, and he explained how only recently that he had learned that the word 'free' had two meanings-free as in 'libre' and free as in 'gratis'-in English."

Michael Tiemann, founder of Cygnus, proposed an alternative to the troublesome "free software" term: sourceware. "Nobody got too excited about it," O'Reilly recalls. "That's when Eric threw out the term 'open source.'"

Although the term appealed to some, support for a change in official terminology was far from unanimous. At the end of the one-day conference, attendees put the three terms--free software, open source, or sourceware--to a vote. According to O'Reilly, 9 out of the 15 attendees voted for "open source." Although some still quibbled with the term, all attendees agreed to use it in future discussions with the press. "We wanted to go out with a solidarity message," O'Reilly says.

The term didn't take long to enter the national lexicon. Shortly after the summit, O'Reilly shepherded summit attendees to a press conference attended by reporters from the [New York Times], the [Wall Street Journal], and other prominent publications. Within a few months, Torvalds' face was appearing on the cover of [Forbes] magazine, with the faces of Stallman, Perl creator Larry Wall, and Apache team leader Brian Behlendorf featured in the interior spread. Open source was open for business.

For summit attendees such as Tiemann, the solidarity message was the most important thing. Although his company had achieved a fair amount of success selling free software tools and services, he sensed the difficulty other programmers and entrepreneurs faced.

"There's no question that the use of the word free was confusing in a lot of situations," Tiemann says. "Open source positioned itself as being business friendly and business sensible. Free software positioned itself as morally righteous. For better or worse we figured it was more advantageous to align with the open source crowd."

For Stallman, the response to the new "open source" term was slow in coming. Raymond says Stallman briefly considered adopting the term, only to discard it. "I know because I had direct personal conversations about it," Raymond says.

By the end of 1998, Stallman had formulated a position: open source, while helpful in communicating the technical advantages of free software, also encouraged speakers to soft-pedal the issue of software freedom. Given this drawback, Stallman would stick with the term free software.

Summing up his position at the 1999 LinuxWorld Convention and Expo, an event billed by Torvalds himself as a "coming out party" for the Linux community, Stallman implored his fellow hackers to resist the lure of easy compromise.

"Because we've shown how much we can do, we don't have to be desperate to work with companies or compromise our goals," Stallman said during a panel discussion. "Let them offer and we'll accept. We don't have to change what we're doing to get them to help us. You can take a single step towards a goal, then another and then more and more and you'll actually reach your goal. Or, you can take a half measure that means you don't ever take another step and you'll never get there."

Even before the LinuxWorld show, however, Stallman was showing an increased willingness to alienate his more conciliatory peers. A few months after the Freeware Summit, O'Reilly hosted its second annual Perl Conference. This time around, Stallman was in attendance. During a panel discussion lauding IBM's decision to employ the free software Apache web server in its commercial offerings, Stallman, taking advantage of an audience microphone, disrupted the proceedings with a tirade against panelist John Ousterhout, creator of the Tcl scripting language. Stallman branded Ousterhout a "parasite" on the free software community for marketing a proprietary version of Tcl via Ousterhout's startup company, Scriptics. "I don't think Scriptics is necessary for the continued existence of Tcl," Stallman said to hisses from the fellow audience members. [6]

"It was a pretty ugly scene," recalls Prime Time Freeware's Rich Morin. "John's done some pretty respectable things: Tcl, Tk, Sprite. He's a real contributor."

Despite his sympathies for Stallman and Stallman's position, Morin felt empathy for those troubled by Stallman's discordant behavior.

Stallman's Perl Conference outburst would momentarily chase off another potential sympathizer, Bruce Perens. In 1998, Eric Raymond proposed launching the Open Source Initiative, or OSI, an organization that would police the use of the term "open source" and provide a definition for companies interested in making their own programs. Raymond recruited Perens to draft the definition. [7]

Perens would later resign from the OSI, expressing regret that the organization had set itself up in opposition to Stallman and the FSF. Still, looking back on the need for a free software definition outside the Free Software Foundation's auspices, Perens understands why other hackers might still feel the need for distance. "I really like and admire Richard," says Perens. "I do think Richard would do his job better if Richard had more balance. That includes going away from free software for a couple of months."

Stallman's monomaniacal energies would do little to counteract the public-relations momentum of open source proponents. In August of 1998, when chip-maker Intel purchased a stake in GNU/Linux vendor Red Hat, an accompanying [New York Times] article described the company as the product of a movement "known alternatively as free software and open source." [8] Six months later, a John Markoff article on Apple Computer was proclaiming the company's adoption of the "open source" Apache server in the article headline. [9]

Such momentum would coincide with the growing momentum of companies that actively embraced the "open source" term. By August of 1999, Red Hat, a company that now eagerly billed itself as "open source," was selling shares on Nasdaq. In December, VA Linux--formerly VA Research--was floating its own IPO to historical effect. Opening at $30 per share, the company's stock price exploded past the $300 mark in initial trading only to settle back down to the $239 level. Shareholders lucky enough to get in at the bottom and stay until the end experienced a 698% increase in paper wealth, a Nasdaq record.

Among those lucky shareholders was Eric Raymond, who, as a company board member since the Mozilla launch, had received 150,000 shares of VA Linux stock. Stunned by the realization that his essay contrasting the Stallman-Torvalds managerial styles had netted him $36 million in potential wealth, Raymond penned a follow-up essay. In it, Raymond mused on the relationship between the hacker ethic and monetary wealth:

> Reporters often ask me these days if I think the open-source community will be corrupted by the influx of big money. I tell them what I believe, which is this: commercial demand for programmers has been so intense for so long that anyone who can be seriously distracted by money is already gone. Our community has been self-selected for caring about other things-accomplishment, pride, artistic passion, and each other. [10]

Whether or not such comments allayed suspicions that Raymond and other open source proponents had simply been in it for the money, they drove home the open source community's ultimate message: all you needed to sell the free software concept is a friendly face and a sensible message. Instead of fighting the marketplace head-on as Stallman had done, Raymond, Torvalds, and other new leaders of the hacker community had adopted a more relaxed approach-ignoring the marketplace in some areas, leveraging it in others. Instead of playing the role of high-school outcasts, they had played the game of

celebrity, magnifying their power in the process.

"On his worst days Richard believes that Linus Torvalds and I conspired to hijack his revolution," Raymond says. "Richard’s rejection of the term open source and his deliberate creation of an ideological fissure in my view comes from an odd mix of idealism and territoriality. There are people out there who think it’s all Richard’s personal ego. I don’t believe that. It’s more that he so personally associates himself with the free software idea that he sees any threat to that as a threat to himself."

Ironically, the success of open source and open source advocates such as Raymond would not diminish Stallman’s role as a leader. If anything, it gave Stallman new followers to convert. Still, the Raymond territoriality charge is a damning one. There are numerous instances of Stallman sticking to his guns more out of habit than out of principle: his initial dismissal of the Linux kernel, for example, and his current unwillingness as a political figure to venture outside the realm of software issues.

Then again, as the recent debate over open source also shows, in instances when Stallman has stuck to his guns, he’s usually found a way to gain ground because of it. "One of Stallman’s primary character traits is the fact he doesn’t budge," says Ian Murdock. "He’ll wait up to a decade for people to come around to his point of view if that’s what it takes."

Murdock, for one, finds that unbudgeable nature both refreshing and valuable. Stallman may no longer be the solitary leader of the free software movement, but he is still the polestar of the free software community. "You always know that he’s going to be consistent in his views," Murdock says. "Most people aren’t like that. Whether you agree with him or not, you really have to respect that."

Notas

[1] See Peter Salus, "FYI-Conference on Freely Redistributable Software, 2/2, Cambridge" (1995) (archived by Terry Winograd). http://hci.stanford.edu/pcd-archives/pcd-fyi/1995/0078.html

[2] Although Linus Torvalds is Finnish, his mother tongue is Swedish. "The Rampantly Unofficial Linus FAQ" offers a brief explanation:

> Finland has a significant (about 6%) Swedish-speaking minority population. They call themselves "finlandssvensk" or "finlandssvenskar" and consider themselves Finns; many of their families have lived in Finland for centuries. Swedish is one of Finland's two official languages.

http://tuxedo.org/~esr/faqs/linus/

[3] Brooks' Law is the shorthand summary of the following quote taken from Brooks' book:

> Since software construction is inherently a systems effort--an exercise in complex interrelationships-communication effort is great, and it quickly dominates the decrease in individual task time brought about by partitioning. Adding more men then lengthens, not shortens, the schedule.

See Fred P. Brooks, [The Mythical Man-Month](Addison Wesley Publishing, 1995)

[4] See Eric Raymond, "The Cathredral and the Bazaar" (1997).

[5] See Malcolm Maclachlan, "Profit Motive Splits Open Source Movement," [TechWeb News] (August 26, 1998). http://content.techweb.com/wire/story/TWB19980824S0012

[6] See Malcolm Maclachlan, "Profit Motive Splits Open Source Movement," [TechWeb News] (August 26, 1998). http://content.techweb.com/wire/story/TWB19980824S0012

[7] See Bruce Perens et al., "The Open Source Definition," The Open Source Initiative (1998). http://www.opensource.org/docs/definition.html

[8] See Amy Harmon, "For Sale: Free Operating System," [New York Times] (September 28, 1998). http://www.nytimes.com/library/tech/98/09/biztech/articles/28linux.html

[9] See John Markoff, "Apple Adopts 'Open Source' for its Server Computers," [New York Times] (March 17, 1999). http://www.nytimes.com/library/tech/99/03/biztech/articles/17apple.html

[10] See Eric Raymond, "Surprised by Wealth," [Linux Today] (December 10, 1999). http://linuxtoday.com/news_story.php3?ltsn=1999-12-10-001-05-NW-LF

Capítulo 12. A Brief Journey Through Hacker Hell

Richard Stallman stares, unblinking, through the windshield of a rental car, waiting for the light to change as we make our way through downtown Kihei.

The two of us are headed to the nearby town of Pa’ia, where we are scheduled to meet up with some software programmers and their wives for dinner in about an hour or so.

It’s about two hours after Stallman’s speech at the Maui High Performance Center, and Kihei, a town that seemed so inviting before the speech, now seems profoundly uncooperative. Like most beach cities, Kihei is a one-dimensional exercise in suburban sprawl. Driving down its main drag, with its endless succession of burger stands, realty agencies, and bikini shops, it’s hard not to feel like a steel-coated morsel passing through the alimentary canal of a giant commercial tapeworm. The feeling is exacerbated by the lack of side roads. With nowhere to go but forward, traffic moves in spring-like lurches. 200 yards ahead, a light turns green. By the time we are moving, the light is yellow again.

For Stallman, a lifetime resident of the east coast, the prospect of spending the better part of a sunny Hawaiian afternoon trapped in slow traffic is enough to trigger an embolism. Even worse is the knowledge that, with just a few quick right turns a quarter mile back, this whole situation easily could have been avoided. Unfortunately, we are at the mercy of the driver ahead of us, a programmer from the lab who knows the way and who has decided to take us to Pa’ia via the scenic route instead of via the nearby Pilani Highway.

"This is terrible," says Stallman between frustrated sighs. "Why didn’t we take the other route?"

Again, the light a quarter mile ahead of us turns green. Again, we creep forward a few more car lengths. This process continues for another 10 minutes, until we finally reach a major crossroad promising access to the adjacent highway.

The driver ahead of us ignores it and continues through the intersection.

"Why isn’t he turning?" moans Stallman, throwing up his hands in frustration. "Can you believe this?"

I decide not to answer either. I find the fact that I am sitting in a car with Stallman in the driver seat, in Maui no less, unbelievable enough. Until two hours ago, I didn’t even know Stallman knew how to drive. Now, listening to Yo-Yo Ma’s cello playing the mournful bass notes of "Appalachian Journey" on the car stereo and watching the sunset pass by on our left, I do my best to fade into the upholstery.

When the next opportunity to turn finally comes up, Stallman hits his right turn signal in an attempt to cue the driver ahead of us. No such luck. Once again, we creep slowly through the intersection, coming to a stop a good 200 yards before the next light. By now, Stallman is livid.

"It’s like he’s deliberately ignoring us," he says, gesturing and pantomiming like an air craft carrier landing-signals officer in a futile attempt to catch our guide’s eye. The guide appears unfazed, and for the next five minutes all we see is a small portion of his head in the rearview mirror.

I look out Stallman's window. Nearby Kahoolawe and Lanai Islands provide an ideal frame for the setting sun. It's a breathtaking view, the kind that makes moments like this a bit more bearable if you're a Hawaiian native, I suppose. I try to direct Stallman's attention to it, but Stallman, by now obsessed by the inattentiveness of the driver ahead of us, blows me off.

When the driver passes through another green light, completely ignoring a "Pilani Highway Next Right," I grit my teeth. I remember an early warning relayed to me by BSD programmer Keith Bostic. "Stallman does not suffer fools gladly," Bostic warned me. "If somebody says or does something stupid, he'll look them in the eye and say, 'That's stupid.'"

Looking at the oblivious driver ahead of us, I realize that it's the stupidity, not the inconvenience, that's killing Stallman right now.

"It's as if he picked this route with absolutely no thought on how to get there efficiently," Stallman says.

The word "efficiently" hangs in the air like a bad odor. Few things irritate the hacker mind more than inefficiency. It was the inefficiency of checking the Xerox laser printer two or three times a day that triggered Stallman's initial inquiry into the printer source code. It was the inefficiency of rewriting software tools hijacked by commercial software vendors that led Stallman to battle Symbolics and to launch the GNU Project. If, as Jean Paul Sartre once opined, hell is other people, hacker hell is duplicating other people's stupid mistakes, and it's no exaggeration to say that Stallman's entire life has been an attempt to save mankind from these fiery depths.

This hell metaphor becomes all the more apparent as we take in the slowly passing scenery. With its multitude of shops, parking lots, and poorly timed street lights, Kihei seems less like a city and more like a poorly designed software program writ large. Instead of rerouting traffic and distributing vehicles through side streets and expressways, city planners have elected to run everything through a single main drag. From a hacker perspective, sitting in a car amidst all this mess is like listening to a CD rendition of nails on a chalkboard at full volume.

"Imperfect systems infuriate hackers," observes Steven Levy, another warning I should have listened to before climbing into the car with Stallman. "This is one reason why hackers generally hate driving cars--the system of randomly programmed red lights and oddly laid out one-way streets causes delays which are so goddamn *unnecessary* [Levy's emphasis] that the impulse is to rearrange signs, open up traffic-light control boxes … redesign the entire system." [1]

More frustrating, however, is the duplicity of our trusted guide. Instead of searching out a clever shortcut--as any true hacker would do on instinct--the driver ahead of us has instead chosen to play along with the city planners' game. Like Virgil in Dante's [Inferno], our guide is determined to give us the full guided tour of this hacker hell whether we want it or not.

Before I can make this observation to Stallman, the driver finally hits his right turn signal. Stallman's hunched shoulders relax slightly, and for a moment the air of tension within the car dissipates. The tension comes back, however, as the driver in front of us slows down. "Construction Ahead" signs line both sides of the street, and even though the Pilani Highway lies less than a quarter mile off in the distance, the two-lane road between us and the highway is blocked by a dormant bulldozer and two large mounds of dirt.

It takes Stallman a few seconds to register what's going on as our guide begins executing a clumsy five-point U-turn in front of us. When he catches a glimpse of the bulldozer and the "No Through Access" signs just beyond, Stallman finally boils over.

"Why, why, why?" he whines, throwing his head back. "You should have known the road was blocked. You should have known this way wouldn’t work. You did this deliberately."

The driver finishes the turn and passes us on the way back toward the main drag. As he does so, he shakes his head and gives us an apologetic shrug. Coupled with a toothy grin, the driver’s gesture reveals a touch of mainlander frustration but is tempered with a protective dose of islander fatalism. Coming through the sealed windows of our rental car, it spells out a succinct message: "Hey, it’s Maui; what are you gonna do?"

Stallman can take it no longer.

"Don’t you fucking smile!" he shouts, fogging up the glass as he does so. "It’s your fucking fault. This all could have been so much easier if we had just done it my way."

Stallman accents the words "my way" by gripping the steering wheel and pulling himself towards it twice. The image of Stallman’s lurching frame is like that of a child throwing a temper tantrum in a car seat, an image further underlined by the tone of Stallman’s voice. Halfway between anger and anguish, Stallman seems to be on the verge of tears.

Fortunately, the tears do not arrive. Like a summer cloudburst, the tantrum ends almost as soon as it begins. After a few whiny gasps, Stallman shifts the car into reverse and begins executing his own U-turn. By the time we are back on the main drag, his face is as impassive as it was when we left the hotel 30 minutes earlier.

It takes less than five minutes to reach the next cross-street. This one offers easy highway access, and within seconds, we are soon speeding off toward Pa’ia at a relaxing rate of speed. The sun that once loomed bright and yellow over Stallman’s left shoulder is now burning a cool orange-red in our rearview mirror. It lends its color to the gauntlet wili wili trees flying past us on both sides of the highway.

For the next 20 minutes, the only sound in our vehicle, aside from the ambient hum of the car’s engine and tires, is the sound of a cello and a violin trio playing the mournful strains of an Appalachian folk tune.

Notas

[1] See Steven Levy, [Hackers] (Penguin USA [paperback], 1984): 40.

Capítulo 13. Continuando con la lucha

Para Richard Stallman, el tiempo puede no curar todas las heridas, pero si es un aliado muy conveniente.

Cuatro años después de "La Catedral y el Bazar", Stallman todavía sigue rumiando la crítica de Raymond. También se queja de la elevación de Linus Torvalds al papel del hacker más famoso del mundo. Para ello recuerda una camiseta muy popular que comenzó a aparecer en las conferencias de Linux alrededor de 1999. Diseñada para imitar el afiche pormocional orginal de la Guerra de las Galaxias, la camiseta muestra a Torvalds blandiendo un sable de luz como Luke Skywalker, mientras que la cara de Stallman se encuentra sobre R2D2. La camiseta aún destroza los nervios de Stallman, no sólo porqué lo ubica e él en el papel de segundón de Torvalds, sino también porqué otorga a Torvalds el papel de líder de la comunidad del software libre/ de fuente abierta, un papel que Torvalds mismo es renuente a aceptar. "Es irónico", dice Stallman lamentándose. "Levantar la espada es exactamente lo que Linus se rehusa a hacer. El logra que todo mundo se concentre en el como el símbolo del movimiento, pero luego no pelea. ¿Para que sirve entonces?

Una vez más, es esa misma renuencia a "levantar la espada", por parte de Torvalds, la que ha dejado la puerta abierta para que Stallman recupere su reputación como el árbitro ético de la comunidad hacker. A pesar de sus lamentos, Stallman tiene que admitir que los últimos años han sido buenos tanto para el como para su organización. Aunque fue relegado a la periferia por el éxito imprevisto de GNU/Linux, Stallman ha sido capaz de recuperar la iniciativa con éxito. Su calendario de charlas entre Enero del 2000 y Diciembre del 2001 incluyó paradas en seis continentes y visitas a países en los que la noción de libertad de software tiene consecuencias importantes --China e India, por ejemplo.

Fuera del púlpito, Stallman también ha aprendido a utilizar su poder como autor y defensor de la Licencia Pública de GNU (más conocida como la GPL (*GNU General Public License*). Durante el verano del 2000, mientras el aire escapaba rápidamente de la burbuja bursátil de Linux de 1999, Stallman y la Fundación para el Software Libre se anotaron dos victorias importantes. En julio del 2000, Troll Tech, una compañia de software noruega, desarrolladora de Qt, un valioso conjunto de herramientas gráficas para el sistema operativo GNU/Linux, anuncio que iba a licenciar su software bajo la GPL. Unas pocas semanas mas tarde, Sun Microsystems, una compañía que hasta entonces había intentado montarse al tren del código de fuente abierta, sin entregar del todo su código, anunció que iba a licenciar de manera doble la nueva aplicación de oficina OpenOffice bajo la Licencia Pública Leve de GNU (*Lesser GNU Public License* o LGPL) y la Licencia de Estándares de la Industria de Sun (*Sun Industry Standards Source License* o SISSL).

Detrás de cada victoria estaba el hecho de que Stallman había hecho poco para pelear por ellas. En el caso de Troll Tech, Stallman simplemente había jugado el papel de pontífice del software libre. En 1999 la compañía había creado una licencia que cumplía las condiciones establecidas por la Fundación para el Software Libre, pero examinandola de manera mas detenida, Stallman detectó incompatibilidades legales que harían imposible incluir Qt con programas protegidos por la licencia GPL. Cansada de enfrentarse a Stallman, la gerencia de Troll Tech finalmente decidió dividir Qt en dos versiones, una protegida por la GPL y otra protegida por la QPL, dándoles a los desarrolladores una manera de rodear los problemas de compatibilidad citados por Stallman.

En el caso de Sun, ellos deseaban jugar de acuerdo a las reglas de la Fundación para el Software libre. En la Conferencia de Open Source de O'Reilly, en 1999, el cofundador y jefe científico de Sun Microsystems, Bill Joy defendió la licencia de "fuente comunitaria" (*community source*) de su compañía, que no era más que un compromiso diluido que permitía a los usuarios copiar y modificar software de propiedad de Sun, impidiéndoles cobrar dicho software sin antes negociar un acuerdo de regalías con Sun. Un año después del discurso de Joy, el vicepresidente de Sun Microsystems, Marco Boerries se encontraba en el mismo escenario explicando el nuevo compromiso de licenciamiento en el caso de OpenOffice, un grupo de aplicaciones de oficina diseñado específicamente para el sistema operativo GNU/Linux.

"Lo puedo deletrear con tres letras", dijo Boerries. "GPL."

En ese momento, Boerries dijo que la decisión de su compañía tenía muy poco que ver con Stallman y estaba mas relacionada con el momentum de los programas protegidos por la GPL. "Básicamente reconocimos que diferentes productos atraen a diferentes comunidades, y la licencia que usemos depende del tipo de comunidad que queramos atraer", dijo Boerries. "Con [OpenOffice], era claro que teníamos la mayor correlación con la comunidad GPL." [1]

Comentarios así muestran la poco reconocida fuerza de la GPL e, indirectamente, el genio político del hombre que desempeñó el papel mas importante en su creación. "No hay ningún abogado sobre la faz de la tierra que hubiera escrito la GPL tal como es", dice Eben Moglen, profesor de leyes de la Universidad de Columbia y consejero general de la Fundación para el Software Libre. "Pero funciona. Y lo hace gracias a la filosofía de diseño de Richard."

El trabajo de Moglen, un antiguo programador profesional, con Stallman se remonta a 1990, cuando Stallman solicitó la asesoría legal de Moglen para un asunto privado. Moglen, quién trabajaba entonces con el experto en encripción Phillip Zimmerman durante las batallas legales de este contra la Administración de Seguridad Nacional, dice que se sintió halagado con la solicitud. "Le dije que usaba Emacs todos los días de mi vida, y que sería necesario mucho trabajo legal de mi parte para pagar la deuda."

Desde entonces Moglen, tal vez más que ningún otro individuo, ha tenido la oportunidad de observar como la filosofía hacker de Stallman se hace presente en el ámbito legal. Moglen dice que las formas en que Stallman se acerca a los códigos legales y al código del software son, en su mayor parte, similares. "Debo decir que, como abogado, pensar que lo que hay que hacer con un documento es corregir todos los errores (*bugs*) no tiene mucho sentido", dice Moglen. "Hay incertidumbre en todo proceso legal, y lo que la mayoría de los abogados quiere hacer es obtener para su cliente los beneficios de esa incertidumbre. El objetivo de Richard es completamente opuesto. Su meta es eliminar la incertidumbre, lo que es inherentemente imposible. Es inherentemente imposible escribir una licencia que controle todas las circunstancias de todos los sistemas legales de todo el mundo. Pero si quisiera intentarlo debería intentarlo en la manera en que Richard lo hace. La elegancia resultante, junto con la simplicidad de diseño del producto, casi logran lo que hay que lograr. Y de allí un poco de trabajo legal puede llevar muy lejos."

Como la persona encargada de llevar el ideario de Stallman, Moglen comprende la frustración de los aliados potenciales. "Richard es un hombre que no quiere comprometer asuntos que el considera fundamentales", dice Moglen, "y no toma a la ligera los cambios de las palabras o incluso la búsqueda de ambigüedad artística, que la sociedad requiere frecuentemente de gran cantidad de gente".

Como la Fundación para el Software Libre no ha querido hacer presencia en asuntos por fuera del desarrollo de GNU y de hacer cumplir la GPL, Moglen ha dedicado su energía sobrante a ayudar a la *Electronic Frontier Foundation* (Fundación para la Frontera Electrónica), una organización que proporciona soporte legal a personas que deben defenderse contra las leyes de derechos de autor, como Dmitri Skylarov. En el año 2000, Moglen también se desempeño como asesor legal directo de un grupo de hackers que fueron procesados por hacer circular el programa de desencripción de DVD llamado deCSS. A pesar del silencio de su cliente principal en ambos casos, Moglen ha aprendido a apreciar el valor de la terquedad de Stallman. "Han habido ocasiones durante estos años en que he ido donde Richard para decirle ‘Tenemos que hacer esto. Tenemos que hacer aquello. Aquí tenemos la situación estratégica. He aquí la siguiente movida. Esto es lo que tenemos que hacer.’ Y la respuesta de Richard siempre ha sido, ‘No tenemos que hacer nada.’ Sólo espera. Lo que deba hacerse se hará."

"¿Y saben qué?" agrega Moglen. "Generalmente, ha estado en lo cierto."

Este tipo de comentarios contradicen lo que Stallman dice de si mismo: "Yo no soy bueno para jugar juegos", dice Stallman, dirigiéndose a sus múltiples críticos ocultos que lo ven como un hábil estratega. "No soy bueno para mirar hacía adelante y anticipar lo que alguien mas va a hacer. Mi manera de hacer las cosas siempre ha sido concentrarme en los cimientos, decir ‘Hagamos los cimientos tan fuertes como podamos hacerlos.’"

La popularidad cada vez mayor de la licencia GPL y su continua fuerza gravitacional son los mejores homenajes a los cimientos creados por Stallman y sus colegas de GNU. Aunque ya no puede llamarse a si mismo "el último hacker verdadero", Stallman puede sin embargo, tomar todo el crédito por la construcción del marco ético del movimiento del software libre. Que los otros programadores modernos se sientan o no confortables trabajando dentro de ese marco, es inmaterial. El hecho de que exista una opción es el legado más grande de Stallman.

Discutir el legado de Stallman en este punto parece un poco prematuro. Stallman, de 48 años en el momento en que se escriben estas líneas, aún tiene varios años para aumentar o disminuir ese legado. De cualquier manera, la naturaleza automática de la conducción del movimiento de software libre hace que sea tentador examinar la vida de Stallman por fuera de las batallas del día a día de la industria del software, y dentro de un marco más histórico.

Para su propio crédito, Stallman rehusa todas las oportunidades para especular. "Nunca he sido capaz de diseñar planes acerca que como será el futuro", dice Stallman, ofreciendo su propio epitafio anticipado. "Yo simplemente dije ‘Voy a pelear. ¿Quién sabe dónde llegaré?’"

No hay ninguna duda de que, en el proceso de cazar peleas, Stallman ha alienado a quienes de otra manera podrían haber sido sus más grandes paladines. También es un testamento a su entereza ética el que incluso sus opositores políticos de antaño aún puedan decir un par de cosas positivas acerca de él cuando son presionados. La tensión entre Stallman el ideólogo y Stallman el genio hacker, sin embargo, lleva a un biógrafo a pensar: ¿Cómo verá la gente a Stallman cuando la propia personalidad de Stallman no este allí para interponerse el camino?

En los borradores iniciales de este libro, yo llamé a esta pregunta la pregunta "de los 100 años". Esperando estimular una visión objetiva de Stallman y de su trabajo, le solicité a varias luminarias de la industria del software que se sacarán a si mismos del marco temporal actual y se pusieran en la posición de un historiador mirando hacía atrás al movimiento del software libre dentro de 100 años. Desde esa perspectiva es fácil observar similitudes entre Stallman y aquellos americanos del pasado, que aunque fueron marginalmente importantes durante su vida, han adquirido una importancia histórica, en relación a su época. Las comparaciones más simples incluyen a Henry David Thoreau, filósofo trascendentalista y autor de [Sobre la Desobediencia Civil], y a John Muir, fundador del Club

Sierra y padre del movimiento ambientalista moderno. También es fácil ver las similitudes con hombres como William Jennings Bryan, conocido como "El Gran Comunero", líder del movimiento populista, enemigo de los monopolios, y un hombre que, a pesar de haber sido poderoso, parece haber caido en la insignificancia histórica.

A pesar de no ser la primera persona en ver el software como propiedad pública, Stallman tiene garantizado un pie de página en libros de historia futuros gracias a la GPL. Sabiendo eso, parece necesario dar un paso atrás y examinar el legado de Richard Stallman por fuera del marco temporal actual. ¿La GPL será algo que los programadores de software usarán en el año 2102, o será algo que ya habrá sido dejado a un lado mucho tiempo atrás? ¿Será el término "software libre" visto como algo tan anticuado como lo es hoy "plata libre" o será visto como una premonición a la luz de los eventos políticos posteriores? [2]

La predicción del futuro es un deporte arriesgado, pero la mayoría de las personas, cuando les fue formulada la pregunta, parecían listos a contestar. "Dentro de cien años, Richard y unas pocas personas más van a merecer mucho más que una nota de pie de página", dice Moglen. "Van a ser considerados como la línea principal de la historia."

Las "pocas personas más" que Moglen postula para capítulos de libros de texto del futuro incluyen a John Gilmore, asesor de Stallman con la GPL y futuro fundador de la *Electronic Frontier Foundation*, y Theodor Hol Nelson, también conocido como Ted Nelson, autor del libro [*Literary Machines*] (Máquinas Literales). Moglen dice que Stallman, Nelson, y Gilmore se destacan, cada uno en su campo como figuras históricas de importancia. Le da crédito a Nelson, conocido por haber acuñado el término "hipertexto", por haber identificado el dilema de la propiedad de la información en la Era Digital. Gilmore y Stallman, a su vez, reciben el crédito por haber identificado los efectos políticos negativos del control de la información, y por haber creado organizaciones --la Fundación para la Frontera Electrónica en el caso de Gilmore y la Fundación para el Software Libre en el caso de Stallman-- para contrarrestar estos efectos. De los dos, sin embargo, Moglen ve las actividades de Stallman como más personales y de una naturaleza menos política.

"Richard fue único porqué para él las implicaciones éticas del software no libre fueron particularmente claras en un momento muy temprano", dice Moglen. "Esto tiene mucho que ver con la personalidad de Richard, a la que muchas personas, al escribir acerca de él, intentarán tratar como epifenómenal, o incluso, como una desventaja en el proyecto de vida de Richard Stallman."

Gilmore, quién describe su inclusión entre el errático Nelson y el irascible Stallman como un "honor ambiguo", secunda sin embargo el argumento de Moglen. Gilmore escribe:

> Mi predicción es que los escritos de Stallman se mantendrán tan bien como lo han hecho los de Thomas Jefferson; él es un escritor bastante claro, y también es claro en sus principios … El que Richard llegue a ser tan influyente como Jefferson dependerá de si las abstracciones que nosotros llamamos "derechos civiles" terminan siendo más importantes dentro de cien años que las abstracciones que llamamos "software" o "restricciones técnicamente impuestas".

Otro elemento del legado de Stallman que no debe ser pasado por alto, es el modelo de desarrollo colaborativo de software en el que el Proyecto GNU fue pionero. A pesar de tener sus fallas de cuando en cuando, el modelo ha sido capaz de evolucionar para convertirse en un estándar dentro de la industria del desarrollo del software. Teniendo todo eso en cuenta, dice Gilmore, este modelo de desarrollo colaborativo puede terminar siendo incluso más importante que el proyecto GNU, la licencia GPL, o cualquier programa de software particular desarrollado por Stallman:

> Antes de que existiera Internet, era bastante difícil colaborar a distancia en el desarrollo de software, incluso entre equipos que se conocen y confían el uno en el otro. Richard fue pionero en el modelo colaborativo de desarrollo de software, particularmente en aquel desarrollado por voluntarios desorganizados que rara vez se reunen unos con otros. Richard no construyó ninguna de las herramientas básicas para hacer esto (el protocolo TCP, las listas de correo, diff y patch, los archivos tar, RCS, CVS o CVS-remoto), pero utilizó aquellas que estaban disponibles para formar grupos sociales de programadores que pudieran colaborar efectivamente.

Lawrence Lessig, profesor de leyes en Stanford y autor del libro, [*The Future of Ideas*] (El Futuro de las ideas), publicado en el 2001 es igualmente *bullish*. Al igual que muchos eruditos en el área legal, Lessig ve la GPL como un terraplen que protege la llamada "comuna digital", la enorme aglomeración de programas de software en poder de la comunidad, y estándares de redes y telecomunicaciones que han impulsado el crecimiento exponencial de Internet durante las últimas tres décadas. En vez de conectar a Stallman con otros pioneros de Internet, hombres como Vannevar Bush, Vinton Cerf, y J.C.R. Licklider, quienes convencieron a otros de entender la tecnología de computadores a una escala más amplia, Lessig ve el impacto de Stallman como algo más personal e introspectivo; en última instancia, como algo único:

> [Stallman] cambio el debate de "es" a "debería". Hizo ver a la gente lo mucho que había en juego, y construyó un medio para llevar estos ideales adelante… No se realmente cómo ubicarlo en el contexto de Cerf o Licklider. La innovación es distinta. No es sólo acerca de cierto tipo de código, o de permitir la Internet. [Es] mucho más acerca de hacer que la gente vea el valor de un cierto tipo de Internet. No creo que haya nadie más en ese campo, antes o después.

No todo el mundo ve el legado de Stallman inscrito en piedra, por supuesto. Eric Raymond, el proponente de *open source* que siente que el papel de liderazgo de Stallman ha disminuido significativamente desde 1996, ve señales ambiguas cuando contempla la bola de cristal del año 2102:

> Creó que los artefactos de Stallman (la GPL, Emacs, GCC) serán vistos como obras revolucionarias, como piedras angulares del mundo de la información. Creó que la historia será menos benevolente con algunas de las teorías a partir de las cuáles RMS trabajó, y nada benevolente con su tendencia personal hacía un comportamiento territorrialista, de líder sectario.

Stallman mismo, a su vez, ve señales ambiguas:

> Lo que la historia diga acerca del Proyecto GNU, dentro de veinte años, dependerá de quién gane la batalla de la libertad para usar el conocimiento público. Si perdemos, será sólo una nota al pie. Si ganamos, no esta garantizado que la gente conozca el papel del sistema operativo GNU --si piensan que el sistema es "Linux", entonces se habrán construido una imagen falsa de lo que sucedió y por qué. Pero incluso si ganamos, la historia que la gente aprenderá dentro de 100 años muy posiblemente dependa de quien domine en la escena política.

En busca de su propia analogía con el siglo diecinueve, Stallman trae a colación la figura de John Brown, el abolicionista militante visto como un héroe a un lado de la línea Mason Dixon y como un orate al otro. [3]

La revuelta de esclavos de John Brown nunca adquirió fuerza, pero durante el juicio subsiguiente él despertó un clamor nacional en pro de la abolición. Durante la Guerra Civil, John Brown era un héroe; 100 años después, y durante la mayoría del siglo veinte, los textos de historia aún enseñaban que estaba loco. Durante la era de segregación legal, cuándo la intolerancia era absurda, los Estados Unidos aceptaron parcialmente la historia que el Sur deseaba contar de si mismo, y los libros de historia decían muchas cosas falsas acerca de la Guerra Civil y los eventos relacionados con ella.

Ese tipo de comparaciones documentan tanto la naturaleza periférica autopercibida del trabajo actual de Stallman, como la naturaleza binaria de su reputación actual. Es difícil imaginarse la reputación de Stallman cayendo a los niveles de infamia a los que cayó la de Brown durante el periodo que siguió a la Reconstrucción. A pesar de ello y de que Stallman, con excepción de las analogías de guerra ocasionales, ha hecho muy poco para inspirar violencia, es fácil imaginar un futuro en el que las ideas de Stallman terminan entre las cenizas. Al convertir la causa del software libre no en un movimiento masivo sino en una colección de batallas privadas contra la fuerza de las tentaciones propietarias, Stallman ha creado una situación imposible de ganar, especialmente para los muchos acólitos con el mismo caracter terco.

Considerándolo de nuevo, puede que ese mismo caracter algún día sea considerado como el legado más duradero de Stallman. Moglen, un observador cercano durante la última década, advierte a aquellos que piensan que la personalidad de Stallman es contraproducente o epifenomenal a los "artefactos" de la vida de Stallman. Sin esa personalidad, dice Moglen, habría muy pocos de esos preciosos artefactos. Citando a Moglen, un antiguo ayudante de la Corte Suprema:

> Mire, el hombre más grande para el que he trabajado fue Thurgood Marshall. Yo sabía lo que lo hacía un buen hombre. Sabía porqué había sido capaz de cambiar el mundo de la manera en que podía. Estaría aventurando demasiado si tratara de hacer una comparación, porqué no podrían ser más distintos. Thurgood Marshall era un hombre en sociedad, representando una sociedad marginada dentro de la sociedad que lo rodeaba, pero aún así, un hombre en sociedad. Sus habilidades eran habilidades sociales. Pero era todo en una pieza, igual que Stallman. A pesar de ser tan diferentes en todos los demás aspectos, la persona con la que más comparo a Thurgood Marshall en ese aspecto, en ser todo de una pieza, compacto, hecho de la sustancia de la que están hechas las estrellas, es Stallman.

En un intento por afianzar esa imagen, Moglen se inspira en un momento que compartieron en la primavera del año 2000. El éxito de la salida a la bolsa de VA Linux aún resonaba en los medios de negocios, y media docena de asuntos relacionados con el software libre se movían entre los medios. Rodeado por un huracán de historias y de situaciones que necesitaban un comentario, Moglen recuerda haberse sentado a almorzar con Stallman sintiéndose como un naúfrago en el ojo de la tormenta. Durante la siguiente hora, según dice él, la conversación giró calmadamente alrededor de un único tema: reforzar la GPL.

"Estabamos allí sentados hablando sobre lo que íbamos a hacer respecto a ciertos problemas en Europa Oriental y sobre lo que haríamos cuando el problema de la propiedad del contenido comenzara a amenazar al software libre", recuerda Moglen. "A medida que hablábamos, pense brevemente acerca del cuadro que veían los transeuntes que pasaban. He aquí, dos pequeños anarquistas barbados, conspirando y planeando los pasos a seguir. Y, por supuesto, Richard arrancando los nudos de su cabello, dejándolos caer en la sopa y comportándose de la manera de siempre. Cualquiera que escuchara nuestra conversación habría pensado que estábamos locos, pero yo sabía: Yo sabía que la revolución estaba allí, en esa mesa. Eso es lo que hace que ocurra. Y ese hombre es la persona haciéndolo ocurrir."

Moglen dice que el momento, más que cualquier otro, le hizo comprender la simplicidad elemental del estilo de Stallman.

"Fue gracioso", recuerda Moglen. "Yo le dije a él, ‘Richard, tú sabes que tú y yo somos los dos tipos que no ganamos dinero con esta revolución.’ Y luego pagué el almuerzo porqué yo sabía que el no tenía el dinero para hacerlo.’"

Notas

[1] Ver Marco Boerries, entrevista con el autor (Julio, 2000).

[2] El término *Free Silver* se refiere a uno de los estandartes de campaña de Jennings en las presidenciales de 1896. La idea del demócrata era abandonar el patrón oro, que mantenía los precios bajos y la inflación controlada, para pasar a respaldar los dólares emitidos por el gobierno con plata, lo que, según Jennings, redundaría en beneficios para los granjeros, que recibirían mas dinero por sus cosechas. (Nota del traductor)

[3] La línea Mason Dixon era considerada la frontera entre los estados esclavistas y los abolicionistas durante la Guerra de Secesión estadounidense.

Anterior
A Brief Journey Through Hacker Hell

Inicio

Siguiente
Epilogue: Crushing Loneliness

Capítulo 14. Epilogue: Crushing Loneliness

Writing the biography of a living person is a bit like producing a play. The drama in front of the curtain often pales in comparison to the drama backstage.

In [The Autobiography of Malcolm X], Alex Haley gives readers a rare glimpse of that backstage drama. Stepping out of the ghostwriter role, Haley delivers the book's epilogue in his own voice. The epilogue explains how a freelance reporter originally dismissed as a "tool" and "spy" by the Nation of Islam spokesperson managed to work through personal and political barriers to get Malcolm X's life story on paper.

While I hesitate to compare this book with [The Autobiography of Malcolm X], I do owe a debt of gratitude to Haley for his candid epilogue. Over the last 12 months, it has served as a sort of instruction manual on how to deal with a biographical subject who has built an entire career on being disagreeable. From the outset, I envisioned closing this biography with a similar epilogue, both as an homage to Haley and as a way to let readers know how this book came to be.

The story behind this story starts in an Oakland apartment, winding its way through the various locales mentioned in the book--Silicon Valley, Maui, Boston, and Cambridge. Ultimately, however, it is a tale of two cities: New York, New York, the book-publishing capital of the world, and Sebastopol, California, the book-publishing capital of Sonoma County.

The story starts in April, 2000. At the time, I was writing stories for the ill-fated BeOpen web site (http://www.beopen.com). One of my first assignments was a phone interview with Richard M. Stallman. The interview went well, so well that Slashdot (http://www.slashdot.org), the popular "news for nerds" site owned by VA Software, Inc. (formerly VA Linux Systems and before that, VA Research), gave it a link in its daily list of feature stories. Within hours, the web servers at BeOpen were heating up as readers clicked over to the site.

For all intents and purposes, the story should have ended there. Three months after the interview, while attending the O'Reilly Open Source Conference in Monterey, California, I received the following email message from Tracy Pattison, foreign-rights manager at a large New York publishing house:

```
To: mailto:sam@BeOpen.com
Subject: RMS InterviewDate: Mon, 10 Jul 2000 15:56:37 -0400

Dear Mr. Williams,

I read your interview with Richard Stallman on BeOpen with great
interest. I've been intrigued by RMS and his work for some time now
and was delighted to find your piece which I really think you did a
great job of capturing some of the spirit of what Stallman is trying
to do with GNU-Linux and the Free Software Foundation.

What I'd love to do, however, is read more - and I don't think I'm
alone. Do you think there is more information and/or sources out there
to expand and update your interview and adapt it into more of a
profile of Stallman? Perhaps including some more anecdotal information
about his personality and background that might really interest and
enlighten readers outside the more hardcore programming scene?
```

The email asked that I give Tracy a call to discuss the idea further. I did just that. Tracy told me her company was launching a new electronic book line, and it wanted stories that appealed to an early-adopter audience. The e-book format was 30,000 words, about 100 pages, and she had pitched her bosses on the idea of profiling a major figure in the hacker community. Her bosses liked the idea, and in the process of searching for interesting people to profile, she had come across my BeOpen interview with Stallman. Hence her email to me.

That's when Tracy asked me: would I be willing to expand the interview into a full-length feature profile?

My answer was instant: yes. Before accepting it, Tracy suggested I put together a story proposal she could show her superiors. Two days later, I sent her a polished proposal. A week later, Tracy sent me a follow up email. Her bosses had given it the green light.

I have to admit, getting Stallman to participate in an e-book project was an afterthought on my part. As a reporter who covered the open source beat, I knew Stallman was a stickler. I'd already received a half dozen emails at that point upbraiding me for the use of "Linux" instead of "GNU/Linux."

Then again, I also knew Stallman was looking for ways to get his message out to the general public. Perhaps if I presented the project to him that way, he would be more receptive. If not, I could always rely upon the copious amounts of documents, interviews, and recorded online conversations Stallman had left lying around the Internet and do an unauthorized biography.

During my research, I came across an essay titled "Freedom--Or Copyright?" Written by Stallman and published in the June, 2000, edition of the MIT Technology Review, the essay blasted e-books for an assortment of software sins. Not only did readers have to use proprietary software programs to read them, Stallman lamented, but the methods used to prevent unauthorized copying were overly harsh. Instead of downloading a transferable HTML or PDF file, readers downloaded an encrypted file. In essence, purchasing an e-book meant purchasing a nontransferable key to unscramble the encrypted content. Any attempt to open a book's content without an authorized key constituted a criminal violation of the Digital Millennium Copyright Act, the 1998 law designed to bolster copyright enforcement on the Internet. Similar penalties held for readers who converted a book's content into an open file format, even if their only intention was to read the book on a different computer in their home. Unlike a normal book, the reader no longer held the right to lend, copy, or resell an e-book. They only had the right to read it on an authorized machine, warned Stallman:

> We still have the same old freedoms in using paper books. But if e-books replace printed books, that exception will do little good. With "electronic ink," which makes it possible to download new text onto an apparently printed piece of paper, even newspapers could become ephemeral. Imagine: no more used book stores; no more lending a book to your friend; no more borrowing one from the public library-no more "leaks" that might give someone a chance to read without paying. (And judging from the ads for Microsoft Reader, no more anonymous purchasing of books either.) This is the world publishers have in mind for us. [1]

Needless to say, the essay caused some concern. Neither Tracy nor I had discussed the software her company would use nor had we discussed the type of copyright that would govern the e-book's usage. I mentioned the Technology Review article and asked if she could give me information on her company's e-book policies. Tracy promised to get back to me.

Eager to get started, I decided to call Stallman anyway and mention the book idea to him. When I did, he expressed immediate interest and immediate concern. "Did you read my essay on e-books?" he asked.

When I told him, yes, I had read the essay and was waiting to hear back from the publisher, Stallman laid out two conditions: he didn't want to lend support to an e-book licensing mechanism he fundamentally opposed, and he didn't want to come off as lending support. "I don't want to participate in anything that makes me look like a hypocrite," he said.

For Stallman, the software issue was secondary to the copyright issue. He said he was willing to ignore whatever software the publisher or its third-party vendors employed just so long as the company specified within the copyright that readers were free to make and distribute verbatim copies of the e-book's content. Stallman pointed to Stephen King's [The Plant] as a possible model. In June, 2000, King announced on his official web site that he was self-publishing [The Plant] in serial form. According to the announcement, the book's total cost would be $13, spread out over a series of $1 installments. As long as at least 75% of the readers paid for each chapter, King promised to continue releasing new installments. By August, the plan seemed to be working, as King had published the first two chapters with a third on the way.

"I'd be willing to accept something like that," Stallman said. "As long as it also permitted verbatim copying."

I forwarded the information to Tracy. Feeling confident that she and I might be able to work out an equitable arrangement, I called up Stallman and set up the first interview for the book. Stallman agreed to the interview without making a second inquiry into the status issue. Shortly after the first interview, I raced to set up a second interview (this one in Kihei), squeezing it in before Stallman headed off on a 14-day vacation to Tahiti.

It was during Stallman's vacation that the bad news came from Tracy. Her company's legal department didn't want to adjust its copyright notice on the e-books. Readers who wanted to make their books transferable would either have to crack the encryption code or convert the book to an open format such as HTML. Either way, the would be breaking the law and facing criminal penalties.

With two fresh interviews under my belt, I didn't see any way to write the book without resorting to the new material. I quickly set up a trip to New York to meet with my agent and with Tracy to see if there was a compromise solution.

When I flew to New York, I met my agent, Henning Guttman. It was our first face-to-face meeting, and Henning seemed pessimistic about our chances of forcing a compromise, at least on the publisher's end. The large, established publishing houses already viewed the e-book format with enough suspicion and weren't in the mood to experiment with copyright language that made it easier for readers to avoid payment. As an agent who specialized in technology books, however, Henning was intrigued by the novel nature of my predicament. I told him about the two interviews I'd already gathered and the promise not to publish the book in a way that made Stallman "look like a hypocrite." Agreeing that I was in an ethical bind, Henning suggested we make that our negotiating point.

Barring that, Henning said, we could always take the carrot-and-stick approach. The carrot would be the publicity that came with publishing an e-book that honored the hacker community's internal ethics. The stick would be the risks associated with publishing an e-book that didn't. Nine months before Dmitri Skylarov became an Internet cause célèbre, we knew it was only a matter of time before an enterprising programmer revealed how to hack e-books. We also knew that a major publishing house releasing an encryption-protected e-book on Richard M. Stallman was the software equivalent of putting "Steal This E-Book" on the cover.

After my meeting with Henning, I put a call into Stallman. Hoping to make the carrot more enticing, I discussed a number of potential compromises. What if the publisher released the book's content under a split license, something similar to what Sun Microsystems had done with Open Office, the free software desktop applications suite? The publisher could then release commercial versions of the e-book under a normal format, taking advantage of all the bells and whistles that went with the e-book software, while releasing the copyable version under a less aesthetically pleasing HTML format.

Stallman told me he didn't mind the split-license idea, but he did dislike the idea of making the freely copyable version inferior to the restricted version. Besides, he said, the idea was too cumbersome. Split licenses worked in the case of Sun's Open Office only because he had no control over the decision making. In this case, Stallman said, he did have a way to control the outcome. He could refuse to cooperate.

I made a few more suggestions with little effect. About the only thing I could get out of Stallman was a concession that the e-book's copyright restrict all forms of file sharing to "noncommercial redistribution."

Before I signed off, Stallman suggested I tell the publisher that I'd promised Stallman that the work would be free. I told Stallman I couldn't agree to that statement but that I did view the book as unfinishable without his cooperation. Seemingly satisfied, Stallman hung up with his usual sign-off line: "Happy hacking."

Henning and I met with Tracy the next day. Tracy said her company was willing to publish copyable excerpts in a unencrypted format but would limit the excerpts to 500 words. Henning informed her that this wouldn't be enough for me to get around my ethical obligation to Stallman. Tracy mentioned her own company's contractual obligation to online vendors such as Amazon.com. Even if the company decided to open up its e-book content this one time, it faced the risk of its partners calling it a breach of contract. Barring a change of heart in the executive suite or on the part of Stallman, the decision was up to me. I could use the interviews and go against my earlier agreement with Stallman, or I could plead journalistic ethics and back out of the verbal agreement to do the book.

Following the meeting, my agent and I relocated to a pub on Third Ave. I used his cell phone to call Stallman, leaving a message when nobody answered. Henning left for a moment, giving me time to collect my thoughts. When he returned, he was holding up the cell phone.

"It’s Stallman," Henning said.

The conversation got off badly from the start. I relayed Tracy’s comment about the publisher’s contractual obligations.

"So," Stallman said bluntly. "Why should I give a damn about their contractual obligations?"

Because asking a major publishing house to risk a legal battle with its vendors over a 30,000 word e-book is a tall order, I suggested.

"Don’t you see?" Stallman said. "That’s exactly why I’m doing this. I want a signal victory. I want them to make a choice between freedom and business as usual."

As the words "signal victory" echoed in my head, I felt my attention wander momentarily to the passing foot traffic on the sidewalk. Coming into the bar, I had been pleased to notice that the location was less than half a block away from the street corner memorialized in the 1976 Ramones song, "53rd and 3rd," a song I always enjoyed playing in my days as a musician. Like the perpetually frustrated street hustler depicted in that song, I could feel things falling apart as quickly as they had come together. The irony was palpable. After weeks of gleefully recording other people’s laments, I found myself in the position of trying to pull off the rarest of feats: a Richard Stallman compromise.

When I continued hemming and hawing, pleading the publisher’s position and revealing my growing sympathy for it, Stallman, like an animal smelling blood, attacked.

"So that’s it? You’re just going to screw me? You’re just going to bend to their will?"

I brought up the issue of a dual-copyright again.

"You mean license," Stallman said curtly.

"Yeah, license. Copyright. Whatever," I said, feeling suddenly like a wounded tuna trailing a rich plume of plasma in the water.

"Aw, why didn’t you just fucking do what I told you to do!" he shouted.

I must have been arguing on behalf of the publisher to the very end, because in my notes I managed to save a final Stallman chestnut: "I don’t care. What they’re doing is evil. I can’t support evil. Good-bye."

As soon as I put the phone down, my agent slid a freshly poured Guinness to me. "I figured you might need this," he said with a laugh. "I could see you shaking there towards the end."

I was indeed shaking. The shaking wouldn’t stop until the Guinness was more than halfway gone. It felt weird, hearing myself characterized as an emissary of "evil." It felt weirder still, knowing that three months before, I was sitting in an Oakland apartment trying to come up with my next story idea. Now, I was sitting in a part of the world I’d only known through rock songs, taking meetings with publishing executives and drinking beer with an agent I’d never even laid eyes on until the day before. It was all too surreal, like watching my life reflected back as a movie montage.

About that time, my internal absurdity meter kicked in. The initial shaking gave way to convulsions of laughter. To my agent, I must have looked like a another fragile author undergoing an untimely emotional breakdown. To me, I was just starting to appreciate the cynical beauty of my situation. Deal or no deal, I already had the makings of a pretty good story. It was only a matter of finding a place to

tell it. When my laughing convulsions finally subsided, I held up my drink in a toast.

"Welcome to the front lines, my friend," I said, clinking pints with my agent. "Might as well enjoy it."

If this story really were a play, here's where it would take a momentary, romantic interlude. Disheartened by the tense nature of our meeting, Tracy invited Henning and I to go out for drinks with her and some of her coworkers. We left the bar on Third Ave., headed down to the East Village, and caught up with Tracy and her friends.

Once there, I spoke with Tracy, careful to avoid shop talk. Our conversation was pleasant, relaxed. Before parting, we agreed to meet the next night. Once again, the conversation was pleasant, so pleasant that the Stallman e-book became almost a distant memory.

When I got back to Oakland, I called around to various journalist friends and acquaintances. I recounted my predicament. Most upbraided me for giving up too much ground to Stallman in the preinterview negotiation. A former j-school professor suggested I ignore Stallman's "hypocrite" comment and just write the story. Reporters who knew of Stallman's media-savviness expressed sympathy but uniformly offered the same response: it's your call.

I decided to put the book on the back burner. Even with the interviews, I wasn't making much progress. Besides, it gave me a chance to speak with Tracy without running things past Henning first. By Christmas we had traded visits: she flying out to the west coast once, me flying out to New York a second time. The day before New Year's Eve, I proposed. Deciding which coast to live on, I picked New York. By February, I packed up my laptop computer and all my research notes related to the Stallman biography, and we winged our way to JFK Airport. Tracy and I were married on May 11. So much for failed book deals.

During the summer, I began to contemplate turning my interview notes into a magazine article. Ethically, I felt in the clear doing so, since the original interview terms said nothing about traditional print media. To be honest, I also felt a bit more comfortable writing about Stallman after eight months of radio silence. Since our telephone conversation in September, I'd only received two emails from Stallman. Both chastised me for using "Linux" instead of "GNU/Linux" in a pair of articles for the web magazine [Upside Today]. Aside from that, I had enjoyed the silence. In June, about a week after the New York University speech, I took a crack at writing a 5,000-word magazine-length story about Stallman. This time, the words flowed. The distance had helped restore my lost sense of emotional perspective, I suppose.

In July, a full year after the original email from Tracy, I got a call from Henning. He told me that O'Reilly & Associates, a publishing house out of Sebastopol, California, was interested in the running the Stallman story as a biography. The news pleased me. Of all the publishing houses in the world, O'Reilly, the same company that had published Eric Raymond's [The Cathedral and the Bazaar], seemed the most sensitive to the issues that had killed the earlier e-book. As a reporter, I had relied heavily on the O'Reilly book [Open Sources] as a historical reference. I also knew that various chapters of the book, including a chapter written by Stallman, had been published with copyright notices that permitted redistribution. Such knowledge would come in handy if the issue of electronic publication ever came up again.

Sure enough, the issue did come up. I learned through Henning that O'Reilly intended to publish the biography both as a book and as part of its new Safari Tech Books Online subscription service. The Safari user license would involve special restrictions,[2] Henning warned, but O'Reilly was willing to allow for a copyright that permitted users to copy and share and the book's text regardless of medium. Basically, as author, I had the choice between two licenses: the Open Publication License or the GNU

Free Documentation License.

I checked out the contents and background of each license. The Open Publication License (OPL)[3] gives readers the right to reproduce and distribute a work, in whole or in part, in any medium "physical or electronic," provided the copied work retains the Open Publication License. It also permits modification of a work, provided certain conditions are met. Finally, the Open Publication License includes a number of options, which, if selected by the author, can limit the creation of "substantively modified" versions or book-form derivatives without prior author approval.

The GNU Free Documentation License (GFDL),[4] meanwhile, permits the copying and distribution of a document in any medium, provided the resulting work carries the same license. It also permits the modification of a document provided certain conditions. Unlike the OPL, however, it does not give authors the option to restrict certain modifications. It also does not give authors the right to reject modifications that might result in a competitive book product. It does require certain forms of front- and back-cover information if a party other than the copyright holder wishes to publish more than 100 copies of a protected work, however.

In the course of researching the licenses, I also made sure to visit the GNU Project web page titled "Various Licenses and Comments About Them."[5] On that page, I found a Stallman critique of the Open Publication License. Stallman's critique related to the creation of modified works and the ability of an author to select either one of the OPL's options to restrict modification. If an author didn't want to select either option, it was better to use the GFDL instead, Stallman noted, since it minimized the risk of the nonselected options popping up in modified versions of a document.

The importance of modification in both licenses was a reflection of their original purpose-namely, to give software-manual owners a chance to improve their manuals and publicize those improvements to the rest of the community. Since my book wasn't a manual, I had little concern about the modification clause in either license. My only concern was giving users the freedom to exchange copies of the book or make copies of the content, the same freedom they would have enjoyed if they purchased a hardcover book. Deeming either license suitable for this purpose, I signed the O'Reilly contract when it came to me.

Still, the notion of unrestricted modification intrigued me. In my early negotiations with Tracy, I had pitched the merits of a GPL-style license for the e-book's content. At worst, I said, the license would guarantee a lot of positive publicity for the e-book. At best, it would encourage readers to participate in the book-writing process. As an author, I was willing to let other people amend my work just so long as my name always got top billing. Besides, it might even be interesting to watch the book evolve. I pictured later editions looking much like online versions of the [Talmud], my original text in a central column surrounded by illuminating, third-party commentary in the margins.

My idea drew inspiration from Project Xanadu (http://www.xanadu.com), the legendary software concept originally conceived by Ted Nelson in 1960. During the O'Reilly Open Source Conference in 1999, I had seen the first demonstration of the project's open source offshoot Udanax and had been wowed by the result. In one demonstration sequence, Udanax displayed a parent document and a derivative work in a similar two-column, plain-text format. With a click of the button, the program introduced lines linking each sentence in the parent to its conceptual offshoot in the derivative. An e-book biography of Richard M. Stallman didn't have to be Udanax-enabled, but given such technological possibilities, why not give users a chance to play around?[6]

When Laurie Petrycki, my editor at O'Reilly, gave me a choice between the OPL or the GFDL, I indulged the fantasy once again. By September of 2001, the month I signed the contract, e-books had become almost a dead topic. Many publishing houses, Tracy's included, were shutting down their e-book imprints for lack of interest. I had to wonder. If these companies had treated e-books not as a form of publication but as a form of community building, would those imprints have survived?

After I signed the contract, I notified Stallman that the book project was back on. I mentioned the choice O'Reilly was giving me between the Open Publication License and the GNU Free Documentation License. I told him I was leaning toward the OPL, if only for the fact I saw no reason to give O'Reilly's competitors a chance to print the same book under a different cover. Stallman wrote back, arguing in favor of the GFDL, noting that O'Reilly had already used it several times in the past. Despite the events of the past year, I suggested a deal. I would choose the GFDL if it gave me the possibility to do more interviews and if Stallman agreed to help O'Reilly publicize the book. Stallman agreed to participate in more interviews but said that his participation in publicity-related events would depend on the content of the book. Viewing this as only fair, I set up an interview for December 17, 2001 in Cambridge.

I set up the interview to coincide with a business trip my wife Tracy was taking to Boston. Two days before leaving, Tracy suggested I invite Stallman out to dinner.

"After all," she said, "he is the one who brought us together."

I sent an email to Stallman, who promptly sent a return email accepting the offer. When I drove up to Boston the next day, I met Tracy at her hotel and hopped the T to head over to MIT. When we got to Tech Square, I found Stallman in the middle of a conversation just as we knocked on the door.

"I hope you don't mind," he said, pulling the door open far enough so that Tracy and I could just barely hear Stallman's conversational counterpart. It was a youngish woman, mid-20s I'd say, named Sarah.

"I took the liberty of inviting somebody else to have dinner with us," Stallman said, matter-of-factly, giving me the same cat-like smile he gave me back in that Palo Alto restaurant.

To be honest, I wasn't too surprised. The news that Stallman had a new female friend had reached me a few weeks before, courtesy of Stallman's mother. "In fact, they both went to Japan last month when Richard went over to accept the Takeda Award," Lippman told me at the time.[7]

On the way over to the restaurant, I learned the circumstances of Sarah and Richard's first meeting. Interestingly, the circumstances were very familiar. Working on her own fictional book, Sarah said she heard about Stallman and what an interesting character he was. She promptly decided to create a character in her book on Stallman and, in the interests of researching the character, set up an interview with Stallman. Things quickly went from there. The two had been dating since the beginning of 2001, she said.

"I really admired the way Richard built up an entire political movement to address an issue of profound personal concern," Sarah said, explaining her attraction to Stallman.

My wife immediately threw back the question: "What was the issue?"

"Crushing loneliness."

During dinner, I let the women do the talking and spent most of the time trying to detect clues as to whether the last 12 months had softened Stallman in any significant way. I didn't see anything to suggest they had. Although more flirtatious than I remembered--a flirtatiousness spoiled somewhat by the number of times Stallman's eyes seemed to fixate on my wife's chest--Stallman retained the same general level of prickliness. At one point, my wife uttered an emphatic "God forbid" only to receive a typical Stallman rebuke.

"I hate to break it to you, but there is no God," Stallman said.

Afterwards, when the dinner was complete and Sarah had departed, Stallman seemed to let his guard down a little. As we walked to a nearby bookstore, he admitted that the last 12 months had dramatically changed his outlook on life. "I thought I was going to be alone forever," he said. "I'm glad I was wrong."

Before parting, Stallman handed me his "pleasure card," a business card listing Stallman's address, phone number, and favorite pastimes ("sharing good books, good food and exotic music and dance") so that I might set up a final interview.

The next day, over another meal of dim sum, Stallman seemed even more lovestruck than the night before. Recalling his debates with Currier House dorm maters over the benefits and drawbacks of an immortality serum, Stallman expressed hope that scientists might some day come up with the key to immortality. "Now that I'm finally starting to have happiness in my life, I want to have more," he said.

When I mentioned Sarah's "crushing loneliness" comment, Stallman failed to see a connection between loneliness on a physical or spiritual level and loneliness on a hacker level. "The impulse to share code is about friendship but friendship at a much lower level," he said. Later, however, when the subject came up again, Stallman did admit that loneliness, or the fear of perpetual loneliness, had played a major role in fueling his determination during the earliest days of the GNU Project.

"My fascination with computers was not a consequence of anything else," he said. "I wouldn't have been less fascinated with computers if I had been popular and all the women flocked to me. However, it's certainly true the experience of feeling I didn't have a home, finding one and losing it, finding another and having it destroyed, affected me deeply. The one I lost was the dorm. The one that was destroyed was the AI Lab. The precariousness of not having any kind of home or community was very powerful. It made me want to fight to get it back."

After the interview, I couldn't help but feel a certain sense of emotional symmetry. Hearing Sarah describe what attracted her to Stallman and hearing Stallman himself describe the emotions that prompted him to take up the free software cause, I was reminded of my own reasons for writing this book. Since July, 2000, I have learned to appreciate both the seductive and the repellent sides of the Richard Stallman persona. Like Eben Moglen before me, I feel that dismissing that persona as epiphenomenal or distracting in relation to the overall free software movement would be a grievous mistake. In many ways the two are so mutually defining as to be indistinguishable.

While I'm sure not every reader feels the same level of affinity for Stallman--indeed, after reading this book, some might feel zero affinity--I'm sure most will agree. Few individuals offer as singular a human portrait as Richard M. Stallman. It is my sincere hope that, with this initial portrait complete and with the help of the GFDL, others will feel a similar urge to add their own perspective to that portrait.

Notas

[1] See "Safari Tech Books Online; Subscriber Agreement: Terms of Service." http://safari.oreilly.com/mainhlp.asp?help=service

[2] See "Safari Tech Books Online; Subscriber Agreement: Terms of Service." http://safari.oreilly.com/mainhlp.asp?help=service

[3] See "The Open Publication License: Draft v1.0" (June 8, 1999). http://opencontent.org/openpub/

[4] See "The GNU Free Documentation License: Version 1.1" (March, 2000). http://www.gnu.org/copyleft/fdl.html

[5] See http://www.gnu.org/philosophy/license-list.html

[6] Anybody willing to "port" this book over to Udanax, the free software version of Xanadu, will receive enthusiastic support from me. To find out more about this intriguing technology, visit http://www.udanax.com.

[7] Alas, I didn't find out about the Takeda Foundation's decision to award Stallman, along with Linus Torvalds and Ken Sakamura, with its first-ever award for "Techno-Entrepreneurial Achievement for Social/Economic Well-Being" until after Stallman had made the trip to Japan to accept the award. For more information about the award and its accompanying $1 million prize, visit the Takeda site, http://www.takeda-foundation.jp.

Apéndice A. Terminology

For the most part, I have chosen to use the term GNU/Linux in reference to the free software operating system and Linux when referring specifically to the kernel that drives the operating system. The most notable exception to this rule comes in Capítulo 9. In the final part of that chapter, I describe the early evolution of Linux as an offshoot of Minix. It is safe to say that during the first two years of the project's development, the operating system Torvalds and his colleagues were working on bore little similarity to the GNU system envisioned by Stallman, even though it gradually began to share key components, such as the GNU C Compiler and the GNU Debugger.

This decision further benefits from the fact that, prior to 1993, Stallman saw little need to insist on credit.

Some might view the decision to use GNU/Linux for later versions of the same operating system as arbitrary. I would like to point out that it was in no way a prerequisite for gaining Stallman's cooperation in the making of this book. I came to it of my own accord, partly because of the operating system's modular nature and the community surrounding it, and partly because of the apolitical nature of the Linux name. Given that this is a biography of Richard Stallman, it seemed inappropriate to define the operating system in apolitical terms.

In the final phases of the book, when it became clear that O'Reilly & Associates would be the book's publisher, Stallman did make it a condition that I use "GNU/Linux" instead of Linux if O'Reilly expected him to provide promotional support for the book after publication. When informed of this, I relayed my earlier decision and left it up to Stallman to judge whether the resulting book met this condition or not. At the time of this writing, I have no idea what Stallman's judgment will be.

A similar situation surrounds the terms "free software" and "open source." Again, I have opted for the more politically laden "free software" term when describing software programs that come with freely copyable and freely modifiable source code. Although more popular, I have chosen to use the term "open source" only when referring to groups and businesses that have championed its usage. But for a few instances, the terms are completely interchangeable, and in making this decision I have followed the advice of Christine Peterson, the person generally credited with coining the term. "The 'free software' term should still be used in circumstances where it works better," Peterson writes. "['Open source'] caught on mainly because a new term was greatly needed, not because it's ideal."

Apéndice B. Hack, Hackers, and Hacking

To understand the full meaning of the word "hacker," it helps to examine the word's etymology over the years.

[The New Hacker Dictionary], an online compendium of software-programmer jargon, officially lists nine different connotations of the word "hack" and a similar number for "hacker." Then again, the same publication also includes an accompanying essay that quotes Phil Agre, an MIT hacker who warns readers not to be fooled by the word's perceived flexibility. "Hack has only one meaning," argues Agre. "An extremely subtle and profound one which defies articulation."

Regardless of the width or narrowness of the definition, most modern hackers trace the word back to MIT, where the term bubbled up as popular item of student jargon in the early 1950s. In 1990 the MIT Museum put together a journal documenting the hacking phenomenon. According to the journal, students who attended the institute during the fifties used the word "hack" the way a modern student might use the word "goof." Hanging a jalopy out a dormitory window was a "hack," but anything harsh or malicious--e.g., egging a rival dorm's windows or defacing a campus statue--fell outside the bounds. Implicit within the definition of "hack" was a spirit of harmless, creative fun.

This spirit would inspire the word's gerund form: "hacking." A 1950s student who spent the better part of the afternoon talking on the phone or dismantling a radio might describe the activity as "hacking." Again, a modern speaker would substitute the verb form of "goof"--"goofing" or "goofing off"--to describe the same activity.

As the 1950s progressed, the word "hack" acquired a sharper, more rebellious edge. The MIT of the 1950s was overly competitive, and hacking emerged as both a reaction to and extension of that competitive culture. Goofs and pranks suddenly became a way to blow off steam, thumb one's nose at campus administration, and indulge creative thinking and behavior stifled by the Institute's rigorous undergraduate curriculum. With its myriad hallways and underground steam tunnels, the Institute offered plenty of exploration opportunities for the student undaunted by locked doors and "No Trespassing" signs. Students began to refer to their off-limits explorations as "tunnel hacking." Above ground, the campus phone system offered similar opportunities. Through casual experimentation and due diligence, students learned how to perform humorous tricks. Drawing inspiration from the more traditional pursuit of tunnel hacking, students quickly dubbed this new activity "phone hacking."

The combined emphasis on creative play and restriction-free exploration would serve as the basis for the future mutations of the hacking term. The first self-described computer hackers of the 1960s MIT campus originated from a late 1950s student group called the Tech Model Railroad Club. A tight clique within the club was the Signals and Power (S&P) Committee--the group behind the railroad club's electrical circuitry system. The system was a sophisticated assortment of relays and switches similar to the kind that controlled the local campus phone system. To control it, a member of the group simply dialed in commands via a connected phone and watched the trains do his bidding.

The nascent electrical engineers responsible for building and maintaining this system saw their activity as similar in spirit to phone hacking. Adopting the hacking term, they began refining it even further. From the S&P hacker point of view, using one less relay to operate a particular stretch of track meant having one more relay for future play. Hacking subtly shifted from a synonym for idle play to a

synonym for idle play that improved the overall performance or efficiency of the club's railroad system at the same time. Soon S&P committee members proudly referred to the entire activity of improving and reshaping the track's underlying circuitry as "hacking" and to the people who did it as "hackers."

Given their affinity for sophisticated electronics--not to mention the traditional MIT-student disregard for closed doors and "No Trespassing" signs--it didn't take long before the hackers caught wind of a new machine on campus. Dubbed the TX-0, the machine was one of the first commercially marketed computers. By the end of the 1950s, the entire S&P clique had migrated en masse over to the TX-0 control room, bringing the spirit of creative play with them. The wide-open realm of computer programming would encourage yet another mutation in etymology. "To hack" no longer meant soldering unusual looking circuits, but cobbling together software programs with little regard to "official" methods or software-writing procedures. It also meant improving the efficiency and speed of already-existing programs that tended to hog up machine resources. True to the word's roots, it also meant writing programs that served no other purpose than to amuse or entertain.

A classic example of this expanded hacking definition is the game Spacewar, the first interactive video game. Developed by MIT hackers in the early 1960s, Spacewar had all the traditional hacking definitions: it was goofy and random, serving little useful purpose other than providing a nightly distraction for the dozen or so hackers who delighted in playing it. From a software perspective, however, it was a monumental testament to innovation of programming skill. It was also completely free. Because hackers had built it for fun, they saw no reason to guard their creation, sharing it extensively with other programmers. By the end of the 1960s, Spacewar had become a favorite diversion for mainframe programmers around the world.

This notion of collective innovation and communal software ownership distanced the act of computer hacking in the 1960s from the tunnel hacking and phone hacking of the 1950s. The latter pursuits tended to be solo or small-group activities. Tunnel and phone hackers relied heavily on campus lore, but the off-limits nature of their activity discouraged the open circulation of new discoveries. Computer hackers, on the other hand, did their work amid a scientific field biased toward collaboration and the rewarding of innovation. Hackers and "official" computer scientists weren't always the best of allies, but in the rapid evolution of the field, the two species of computer programmer evolved a cooperative--some might say symbiotic--relationship.

It is a testament to the original computer hackers' prodigious skill that later programmers, including Richard M. Stallman, aspired to wear the same hacker mantle. By the mid to late 1970s, the term "hacker" had acquired elite connotations. In a general sense, a computer hacker was any person who wrote software code for the sake of writing software code. In the particular sense, however, it was a testament to programming skill. Like the term "artist," the meaning carried tribal overtones. To describe a fellow programmer as hacker was a sign of respect. To describe oneself as a hacker was a sign of immense personal confidence. Either way, the original looseness of the computer-hacker appellation diminished as computers became more common.

As the definition tightened, "computer" hacking acquired additional semantic overtones. To be a hacker, a person had to do more than write interesting software; a person had to belong to the hacker "culture" and honor its traditions the same way a medieval wine maker might pledge membership to a vintners' guild. The social structure wasn't as rigidly outlined as that of a guild, but hackers at elite institutions such as MIT, Stanford, and Carnegie Mellon began to speak openly of a "hacker ethic": the yet-unwritten rules that governed a hacker's day-to-day behavior. In the 1984 book [Hackers], author Steven Levy, after much research and consultation, codified the hacker ethic as five core hacker tenets.

In many ways, the core tenets listed by Levy continue to define the culture of computer hacking. Still, the guild-like image of the hacker community was undermined by the overwhelmingly populist bias of the software industry. By the early 1980s, computers were popping up everywhere, and programmers who once would have had to travel to top-rank institutions or businesses just to gain access to a machine suddenly had the ability to rub elbows with major-league hackers via the ARPAnet. The more these programmers rubbed elbows, the more they began to appropriate the anarchic philosophies of the hacker culture in places like MIT. Lost within the cultural transfer, however, was the native MIT cultural taboo against malicious behavior. As younger programmers began employing their computer skills to harmful ends-creating and disseminating computer viruses, breaking into military computer systems, deliberately causing machines such as MIT Oz, a popular ARPAnet gateway, to crash--the term "hacker" acquired a punk, nihilistic edge. When police and businesses began tracing computer-related crimes back to a few renegade programmers who cited convenient portions of the hacking ethic in defense of their activities, the word "hacker" began appearing in newspapers and magazine stories in a negative light. Although books like [Hackers] did much to document the original spirit of exploration that gave rise to the hacking culture, for most news reporters, "computer hacker" became a synonym for "electronic burglar."

Although hackers have railed against this perceived misusage for nearly two decades, the term’s rebellious connotations dating back to the 1950s make it hard to discern the 15-year-old writing software programs that circumvent modern encryption programs from the 1960s college student, picking locks and battering down doors to gain access to the lone, office computer terminal. One person’s creative subversion of authority is another person’s security headache, after all. Even so, the central taboo against malicious or deliberately harmful behavior remains strong enough that most hackers prefer to use the term "cracker"--i.e., a person who deliberately cracks a computer security system to steal or vandalize data-to describe the subset of hackers who apply their computing skills maliciously.

This central taboo against maliciousness remains the primary cultural link between the notion of hacking in the early 21st century and hacking in the 1950s. It is important to note that, as the idea of computer hacking has evolved over the last four decades, the original notion of hacking--i.e., performing pranks or exploring underground tunnels-remains intact. In the fall of 2000, the MIT Museum paid tradition to the Institute’s age-old hacking tradition with a dedicated exhibit, the Hall of Hacks. The exhibit includes a number of photographs dating back to the 1920s, including one involving a mock police cruiser. In 1993, students paid homage to the original MIT notion of hacking by placing the same police cruiser, lights flashing, atop the Institute’s main dome. The cruiser’s vanity license plate read IHTFP, a popular MIT acronym with many meanings. The most noteworthy version, itself dating back to the pressure-filled world of MIT student life in the 1950s, is "I hate this fucking place." In 1990, however, the Museum used the acronym as a basis for a journal on the history of hacks. Titled, The Institute for Hacks Tomfoolery and Pranks, the journal offers an adept summary of the hacking.

"In the culture of hacking, an elegant, simple creation is as highly valued as it is in pure science," writes [Boston Globe] reporter Randolph Ryan in a 1993 article attached to the police car exhibit. "A Hack differs from the ordinary college prank in that the event usually requires careful planning, engineering and finesse, and has an underlying wit and inventiveness," Ryan writes. "The unwritten rule holds that a hack should be good-natured, non-destructive and safe. In fact, hackers sometimes assist in dismantling their own handiwork."

The urge to confine the culture of computer hacking within the same ethical boundaries is well-meaning but impossible. Although most software hacks aspire to the same spirit of elegance and simplicity, the software medium offers less chance for reversibility. Dismantling a police cruiser is

easy compared with dismantling an idea, especially an idea whose time has come. Hence the growing distinction between "black hat" and "white hat"--i.e., hackers who turn new ideas toward destructive, malicious ends versus hackers who turn new ideas toward positive or, at the very least, informative ends.

Once a vague item of obscure student jargon, the word "hacker" has become a linguistic billiard ball, subject to political spin and ethical nuances. Perhaps this is why so many hackers and journalists enjoy using it. Where that ball bounces next, however, is anybody’s guess.

Apéndice C. GNU Free Documentation License

Version 1.1, March 2000

1. PREAMBLE

   The purpose of this License is to make a manual, textbook, or other written document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

   This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

   We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

2. APPLICABILITY AND DEFINITIONS

   This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

   A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

   A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

   The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ascii without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

3. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

4. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

5. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document’s license notice.

H. Include an unaltered copy of this License.

I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. In any section entitled "Acknowledgments" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgments and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

6. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections

in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgments", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

7. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

8. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document’s Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

9. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

10. TERMINATION

    You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

11. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

Anterior
Hack, Hackers, and Hacking

Inicio

Siguiente
ADDENDUM: How to use this License for your documents

C.1. ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C) year your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.1 or
any later version published by the Free Software Foundation; with the
Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the
Back-Cover Texts being list.  A copy of the
license is included in the section entitled ‘‘GNU Free Documentation
License’’.
```

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being `list`"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.